(continued from part 2)

Logic circuits and binary bits

In chapter 1 you saw how solid state technology, and in particular integrated circuits, have played an important role in the development of the computer. A single integrated circuit may have to contain an enormous number of logic circuits, but the fact that a large number of the circuits are alike makes this technology simpler to accomplish. Circuits that realise logic functions such as A AND B, A OR B, NOT A and NOT B, by giving them pre-determined values, are used time and time again in computer hardware.

Figure 6a shows the symbol used to represent one such circuit, an AND gate. Electronically it operates with a power supply of + 5 V and electrical signals on the inputs and output that vary between + 2.4 V and 0 V.

Figure 6b shows how the electrical signals on the inputs A or B cause the output signal Q to change. To make the explanation clearer the input changes are made at set times (t_0 to t_8). For example: t_2: A and B are 2.4 V; Q goes to 2.4 V
t_5: A is 0 V; B is 2.4V V; Q stays at 0 V
t_8: A and B are 0 V; Q stays at 0 V

From the signal diagram in *figure 6b* it is obvious that the signals have only two operational levels either 0 V or 2.4 V. To make it easier keeping track of the levels the symbol 1 is assigned to the 2.4 V level and 0 to the 0 V level. The logic state of each signal can easily be identified at any time by drawing a table like the one shown in *figure 6c*.

Looking at *figure 6c* it is easy to see that Q is only 1 when A AND B are both 1 at the same time. That is why this particular logic circuit is called an AND gate, because it performs the logical AND function electrically. Since the signal can be in only one of two states, it is called a **binary circuit.** Only one bit of information, which may be 0 or 1, can be represented by this circuit.

A **truth table** is used to illustrate how a combination of binary or logic input signals (A and B) results in a logical or binary output (Q), for a specific type of gate. *Figure 7* shows truth tables for AND, OR and NOT gates.

The **OR gate** differs from the AND gate in that it needs just *one* input to be 1 (2.4 V) to cause the output to be a logical1 (2.4V) also. Q will be 0 only when both the inputs are 0. The **NOT gate** simply inverts the signal. When an input is 1 the output is 0 and vice versa.

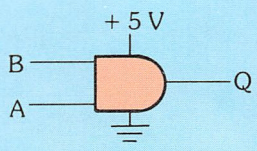
A computer is made up of thousands, often millions of circuits like these. Such circuits handle the digital information in digital codes, detect the codes and execute the operations they specify. The codes are called **machine instructions** or **machine language.**

6a. AND gate logic circuit, standard symbol. **b. timing diagram** and voltage levels of AND gate. **c. logic levels** of bits related to timing and voltage levels in b.

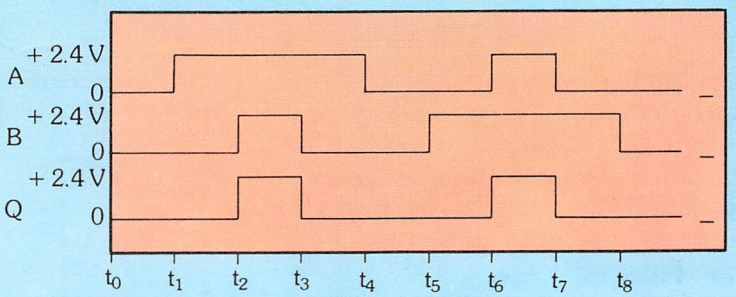
7. Truth tables relating to AND, OR and NOT (inverter) gates.

6

a. AND gate logic circuit (symbolic)



b. Timing diagram and voltage levels of AND gate

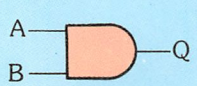


c. Logic levels of bits related to timing and voltage levels in **b**

A	0	1	1	1	0	0	1	0	0
B	0	0	1	0	0	1	1	1	0
Q	0	0	1	0	0	0	1	0	0

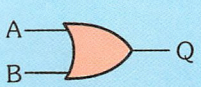
7

AND



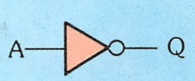
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

OR



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

NOT (inverter)



A	Q
0	1
1	0

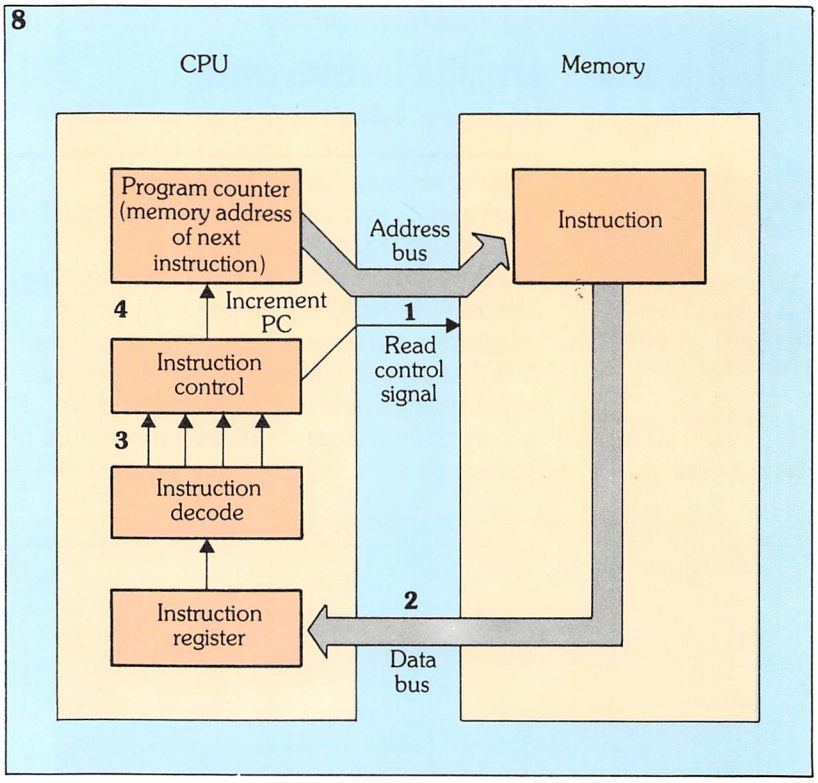
The CPU and its components

The CPU is responsible for the execution and control of system operations. Although most computers have a single CPU some do exist with more than one, and they are said to have a **multi-processing capability.** This means that they can have two processors, both operating on different instructions at the same time. A computer with just one CPU can only carry out one instruction at a time.

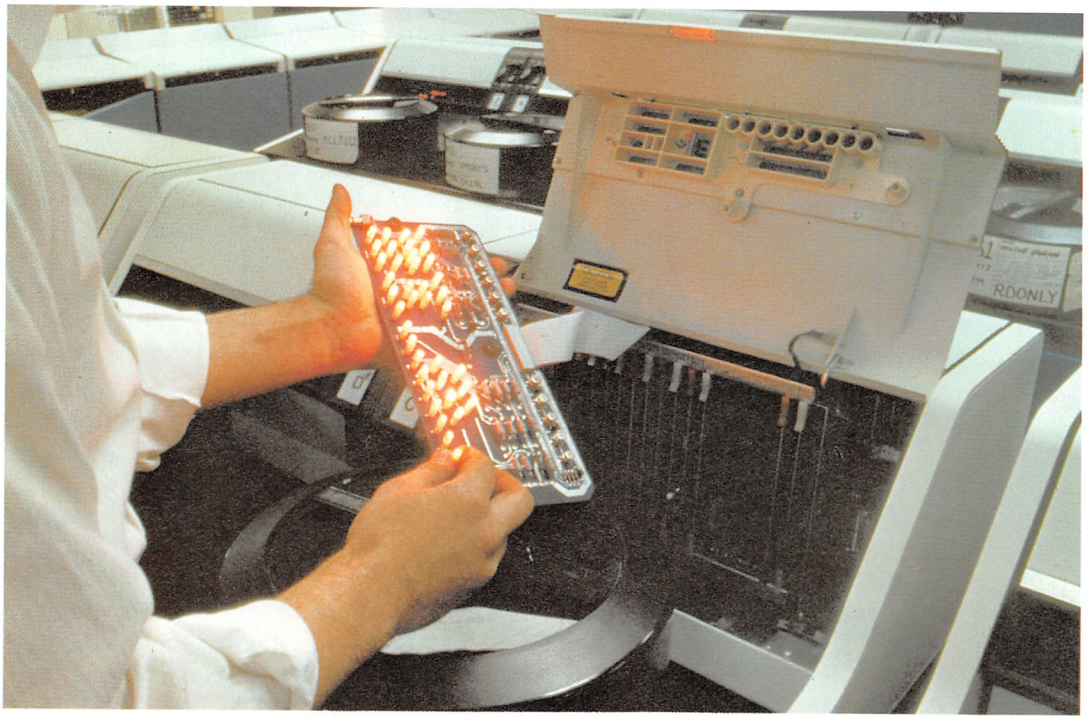
In working through a machine instruction the computer carries out the following operations:

- fetches the instruction from a memory location, specified by an internal register called the program counter
- decodes the instruction and carries out the action specified
- increments (adds one) to the program counter to give the address for the next instruction in sequence (unless a change in operating sequence is indicated).

This sequence of operations is shown in *figure 8.* The address of the next instruction to be fetched is put on the address bus and the read control signal is activated (1). The next instruction is then sent to the CPU along the data bus (2) and is placed in the instruction register. The instruction decode circuitry enables the appropriate control signals to execute the instruction (3). The program counter is incremented (4) so that the next instruction in the sequence can be fetched and the process repeated.



8. The sequence of operations carried out by a computer on receiving a machine instruction.

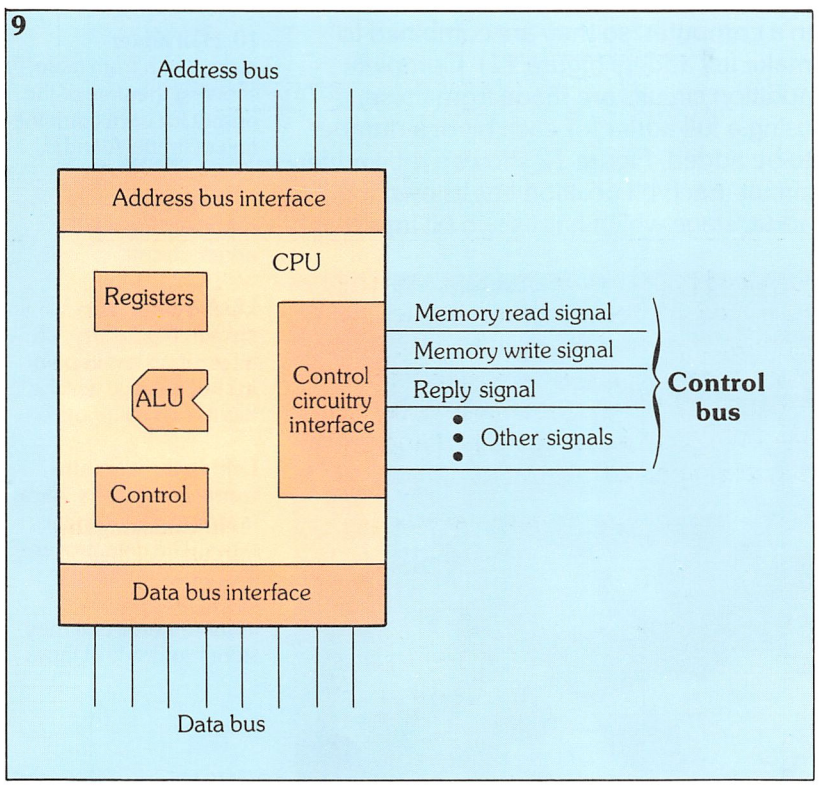


Tony Stone photo library

Left: Checking the circuitry of the indicator panel in a magnetic disk drive.

Chapter 3 will cover computer instructions in more detail. For the moment, the main points to bear in mind are: instructions, contained in the computer's memory, operate on data; executing an instruction involves fetching data from memory or, more often, an exchange of data between the CPU/memory or CPU/external devices; this in turn involves the system's busses in the transfer of data and control signals.

9. The Central Processing Unit (CPU) and its interface with the control, data and address busses. The control circuits ensure that signals arrive at their relevant destinations.



Registers

A register can be a memory location where a piece of data is kept, or one of the registers used by the CPU in carrying out the different operations.

One of the big differences between one CPU and another is the number and type of registers. A register can be part of the central memory, or part of the CPU itself. Contents of registers within the CPU can be obtained much faster than from memory or external storage, thus speeding up the computing process. The register for the program counter is included in the CPU. This keeps track of where the next instruction is located in the memory.

A special memory in the CPU is called the **status register.** This is normally made up of single bit indicators (**flags**). The status register keeps track of the operations in the CPU during the execution of the program. Most processors have at least four flags:
Zero flag: is the result equal to zero?
Sign flag: is the number plus or minus?
Carry flag: did a carry occur?
Parity flag: check whether the total number of bits is odd or even.

Many processors have other status flags but it is not necessary to describe them here.

Control

The **control unit** of the CPU is dedicated to managing the whole system; it carries out all the control functions.
Figure 9 shows the CPU and its interface with the control, data and address busses. As explained earlier, the address bus is used to carry the memory addresses and I/O device addresses so that the appropriate **operand** (specific piece of information for a given operation) is selected. The operand is sent on the data bus. The control circuits ensure that the various signals arrive at their relevant destinations, and that the data necessary to carry out operations is transferred.

One control signal, for example, is used to direct the appropriate memory unit to send the contents of the addressed location to the CPU. This signal is called the **memory read signal.** Another signal on the control bus is used by the selected memory unit to tell the CPU that the data it has requested has been placed on the data bus. This signal is shown in *figure 9* as the **reply signal.** Other control signals are used to control memory writes, input/output operations and so on.

Arithmetic and Logic Unit (ALU)

Arithmetical and logical operations – which form the main body of the computer's work – are carried out in a part of the CPU that is usually treated as a separate unit. This is the **Arithmetic and Logic Unit (ALU).** Data to be used in these calculations is stored in the computer memory as a sequence of binary numbers.

An add operation can be carried out using a half-adder circuit which is made of

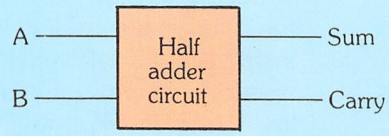
logic gates. Suppose we want to add two one-bit numbers A and B. Because these are one-bit numbers, each can represent only 0 or 1.

The truth table for the half adder circuit (see *figure 10*) shows the state of the output for each combination of inputs A and B. The half-adder has two outputs, one represents SUM and one represents CARRY.

Remember that binary addition works in a different way to 'ordinary' arithmetic. (See *Digital Electronics – 1* for a detailed explanation). If both A and B are equal to one, the result of their addition is equal to two. This is represented by the binary number 10. Just as in ordinary arithmetic, a 'CARRY' function is necessary. The number two (10) is indicated by a binary 1 in the CARRY output and a 0 in the SUM output. The CARRY output of our half adder would be used as an input to the next part of an adder circuit.

Half adders are not particularly useful in a computer, so they are combined to make full adders (*figure 11*). Complete addition circuits are made from these, using a full adder for each bit of a number to be added. *Figure 12* shows a 4 bit adder circuit. Each bit position has its own full adder stage which has its two bit inputs and

10



Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

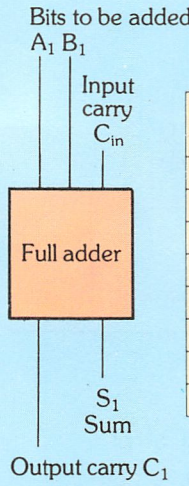
10. Half adder circuit with truth table showing the state of the output for each combination of inputs A and B.

11. Diagram representing a full adder circuit.

12. A 4-bit adder circuit. Note that each bit position has its own full adder stage which has its two bit inputs.

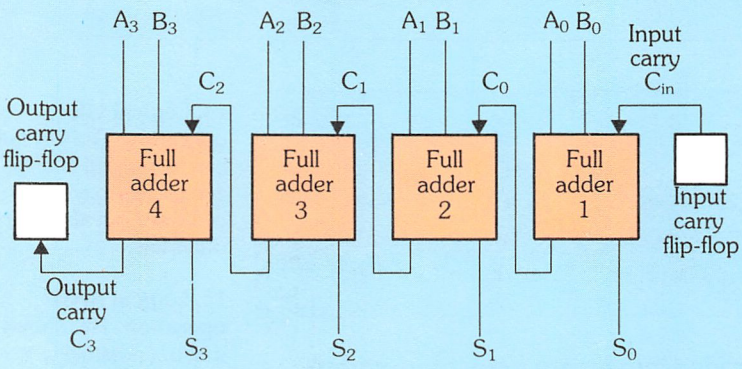
Left: In large, mainframe computers, data is stored on magnetic tapes. The data processing department of a large company or administrative unit may store hundreds of tapes.

11



Inputs			Outputs	
A_1	B_1	C_{in}	S_1	C_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

12

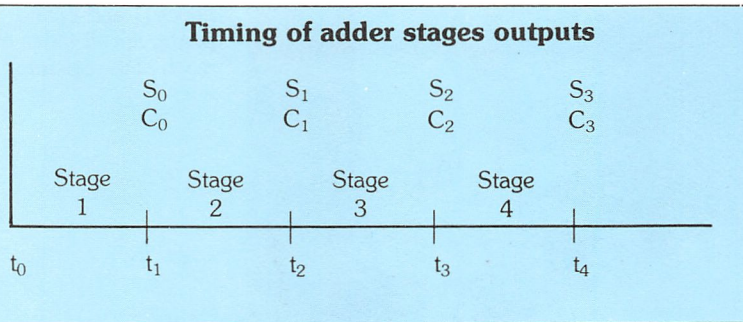


also a carry input from the preceding stage.

Assume that we are using this circuit to add two 4 bit numbers, A and B. This is done in stages. *Figure 13* shows the timing of the adder stages outputs. The least significant (first) bit of A (A_o) is added to the least significant (first) bit of B (B_o) giving a sum (S_o) and a carry (C_o) at time t_1. Next the carry output from stage 1 (C_o) is added along with the second bits of A and B (A_1 and B_1) to give us S_1 and C_1 at time t_2. This process continues through all the stages.

13. Diagram showing the timing of adder stages outputs.

13



The sum outputs go to the **accumulator register** and the last carry bit goes to a special storage position. The CPU reads the value of the two numbers added from the accumulator register and the carry bit.

It may seem odd to say that we can subtract by using the adder but this is quite a simple process. The system we use is to take the number which we wish to subtract and first calculate what is known as its two's complement (inverse +1). Let us assume that the number we wish to subtract is 5, which is written in binary code as 0101. To obtain the two's complement of 5 we invert the binary number (i.e. 0 becomes 1 and vice versa). This gives us 1010. We then add 1 to the figure, giving us 1011. So that 1011 is the two's complement of 5. Now if we wish to subtract 5 from 9 we add the two's complement of 5 to the binary code for 9 and ignore the final carry.

```
 9    1001
-5   +1011
 -   -----
4 (1)0100
```

As you can see the answer is 0100 which is the binary code for 4.

Multiplication can be achieved by repetitive adding i.e. A x B = A + A + A + A + . . . A, B times. Division is achieved by repetitive subtractions i.e. B ÷ A = B – A – A – A – A . . . – A, until the remainder is zero or a number smaller than A. There are in fact more efficient but complex ways of multiplying and dividing, based on similar principles, which will be explained in more detail in a later chapter.

Logical instructions

Having explained in simple terms the functions of OR, AND and NOT circuits, we will now look more closely at the variety of ways in which they can be used.

Each logical operation is done on a bit by bit basis. In the following descriptions we have assumed that each is performed with the input fed into a 2 input gate, each bit position at a time (*figure 7*).

For an OR operation the result is a 1 if either of the input bits are 1. For the NOT operation the result is 1 when the input bits

are 0. For the AND operation the resulting bit is only 1 when both input bits are 1.

Another useful logical operation is the **exclusive OR** (normally written XOR). This will give a result of 1 when *only one* of the inputs is 1. (Note that this is different to the normal OR, which can give a result of 1 if both or either input is 1.)

The OR operation can be used to change certain bits in a binary code without affecting other bits. For example, if the present contents of register A (in *figure 14*) are as shown, we can OR them with the binary number 00001001, to change bits 0 and 3 to 1 without changing the other bits.

Figure 15 shows how a certain bit (in this case bit 0) can be isolated by masking the other bits with an AND operation.

The XOR operation can be used to compare the bits in two binary numbers to see if these numbers are equal. If they are equal, the output will be all zeros. For each pair of bits that are different, the output is one. In *Figure 16* you can see this process demonstrated.

In a computer, all these logical operations have more complicated applications, but these are mainly based on variations of the examples given.

The ALU, which is the main processing unit of the CPU, can easily accomplish all forms of arithmetic and logical operations using combinations of AND, OR and NOT gates, once the data has been decoded into binary form. Even complicated operations such as finding sines and cosines can be done easily using the standard logic elements.

14

Bit no.	7	6	5	4	3	2	1	0
Register A before OR	1	1	1	1	0	0	0	0
OR with	0	0	0	0	1	0	0	1
Register A after OR	1	1	1	1	1	0	0	1

15

Bit no.	7	6	5	4	3	2	1	0
Register A before AND	1	0	1	0	1	1	0	1
AND with	0	0	0	0	0	0	0	1
Register A after AND	0	0	0	0	0	0	0	1

16

	Numbers alike	Numbers not alike
Number A	1101	1101
Number B	1101	1001
XOR results	0000	0100

14. Logical OR operation. The OR ratio can be used to change certain bits in a binary code without affecting other bits.

15. Logical AND operation. This shows that it is possible to isolate a certain bit (bit 0) by masking the other bits with an AND operation.

16. Logical XOR operation. This can be used to compare the bits in two binary numbers to see if these numbers are equal.

Left: In many offices today, the minicomputer is becoming a familiar sight. Keyboard, visual display unit (VDU), disk drives (1 and 2) and printer are all within easy reach of the operator.

Central memory

The memory units of a computer system are used for storing information and instructions. Early memory units were very costly and as a consequence the first computers had very small memories. A memory of 16 K or 32 K (generally in computing terms K = 2^{10} = 1024) was considered large. Today, most microcomputers can easily accept up to 64 K memories, and in the bigger computers memories of millions of cells are common.

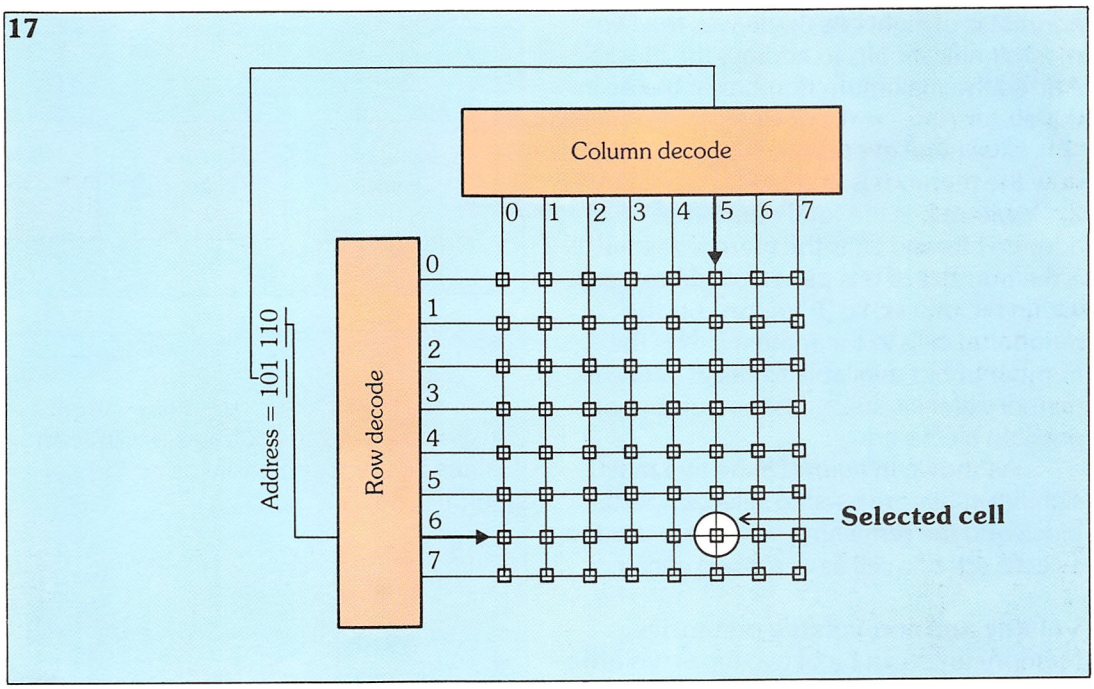
Memory cells are composed of a number of bits. Because the two most common digital codes for representing alphanumeric characters (ASCII and EBCDIC) are easily maintained in 8 bits, most computers are structured so that the smallest addressable cell consists of eight bits. Eight bits are referred to as a byte.

By addressable, we mean a cell has a unique address which locates all the bits in its contents. If there are N cells (e.g N = 1000), and each cell contains a byte, then the addressable units consist of 0 to N – 1 bytes (0-999). If an address consists of M (e.g. M = 12) bits, then the addressing capability of the largest number of cells which can be uniquely selected is 2^M (2^{12} = 4096).

This range is called the **address space** of the computer although it doesn't need to have actual memory cells for all of these locations. For instance, most small microcomputers have an address space of 64K bytes; however, if the application only calls for, say – 4K bytes, only 4K of memory need be included and the other 60K of address space is unused.

Memory cells are often specified in terms of **words.** A word may consist of one or more bytes. In general the word size relates to the registers in the CPU. e.g. if the registers are designed to work on 8-bit codes, then the word size will be 1 byte (8 bits). Words which need to be several bytes long are normally multiples of the basic cell size. For instance, the word may be 4 bytes long and represented in memory by 4 cells of 8 bits each.

Normally memories are organised in a square matrix (see *figure 17*) with an equal number of columns (along the vertical) and rows (going horizontally). Each column and row intersection represents one cell. *Figure 17* shows how a specific cell can be located in the matrix. We have assumed that this cell's address is composed of six bits (101110). After decoding, the three least significant bits are used to specify one of the eight rows; the three most significant bits are used to specify one of the eight columns. The intersection of



17. Memories are usually organised in a square matrix with equal numbers of columns and rows. Here, we see how a single cell is located within the matrix.

the selected column and row then points to the specific memory cell. **Accessing** is the name given to this process of obtaining information from central or peripheral memory.

In reality, there is normally more than one block of memory in a computer. In that case, part of the memory address is used to select a specific block, and the remaining part is used to address a cell within the matrix as described above. The total amount of addressable memory is still 2^M as explained before. But now it is necessary to define how big the matrix is and how many blocks there are.

Figure 18 shows a system that has an eight-bit address. The size of the matrix in this system is 64 cells so six bits are needed to address each cell ($2^6 = 64$). Since the address is of eight bits this leaves the two most significant bits to address the blocks. And so the maximum number of blocks in this system is $2^2 = 4$.

A general expression that describes how the memory is organised is given by 2^{L-N}, where L is the total number of address bits and 2^L is the memory space. N is the number of bits used to address a cell within the matrix, so 2^N represents the number of cells in the matrix; L-N is the number of bits available to select between memory blocks, so 2^{L-N} is the number of possible blocks.

As shown in *figure 18* the two most significant bits are used to select a specific block and the remaining six bits are used to select a specific cell as described earlier.

Volatile and non-volatile memories

Memory units can be of two types, **volatile** and **non-volatile. Volatile** memories are those which lose the information stored in them when the power is turned off. Think of a pocket calculator which has enough memory to store the results of one calculation. When the calculator is switched off, the result is lost because the semiconductor memory used is volatile. Most small computers use volatile semiconductor memories.

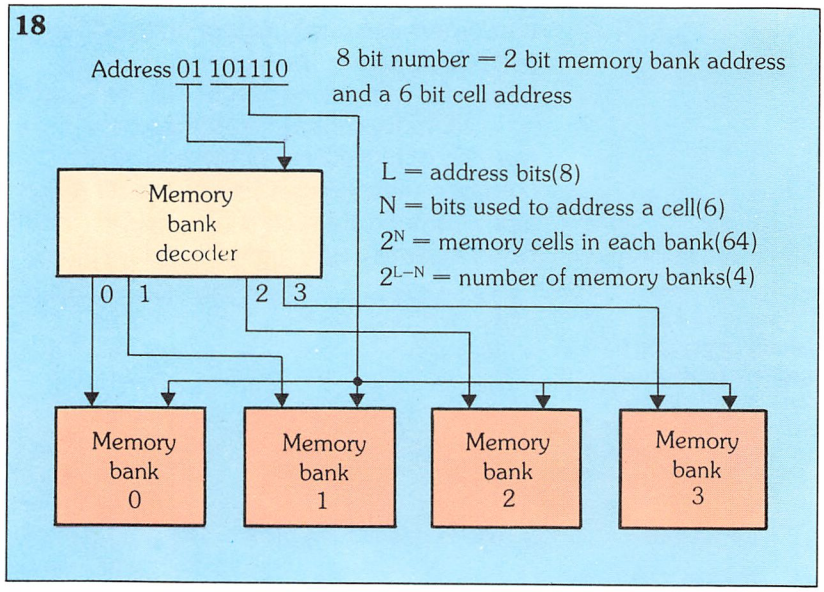
Figure 19 shows the typical circuit for a volatile semiconductor memory. It shows two inverters cross-coupled to hold a bit. This circuit is called a **flip-flop** (it flips in one direction and flops back). If a 0 is put on to the input of the first inverter (S_1), then its output (S_2) will be 1. This then puts the input of the second inverter to 1, which in turn puts its output to 0. Since this output is connected to the input of the first inverter, the first inverter is held in a stable state.

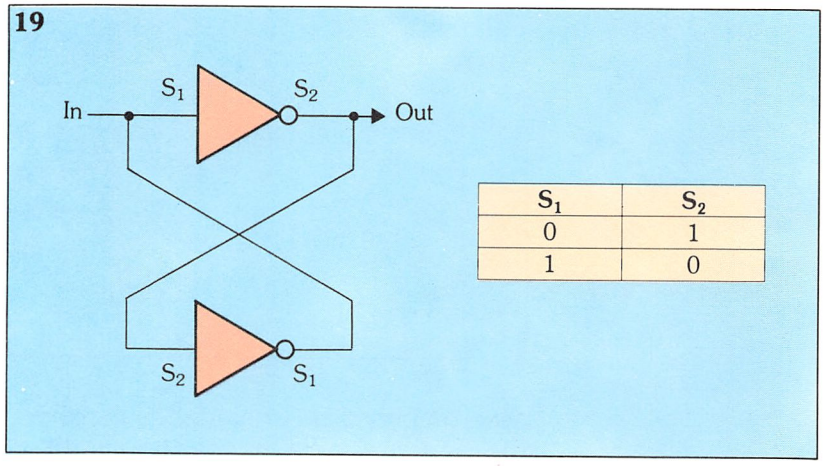
If S_1 is 1, then the output of the second inverter is also 1, once again holding the flip-flop in a stable state. The states of S_1 and S_2 will remain at 0 or 1 as set until they are intentionally changed or until the power is turned off.

Non-volatile memories do not lose the information stored in them when the power is turned off.

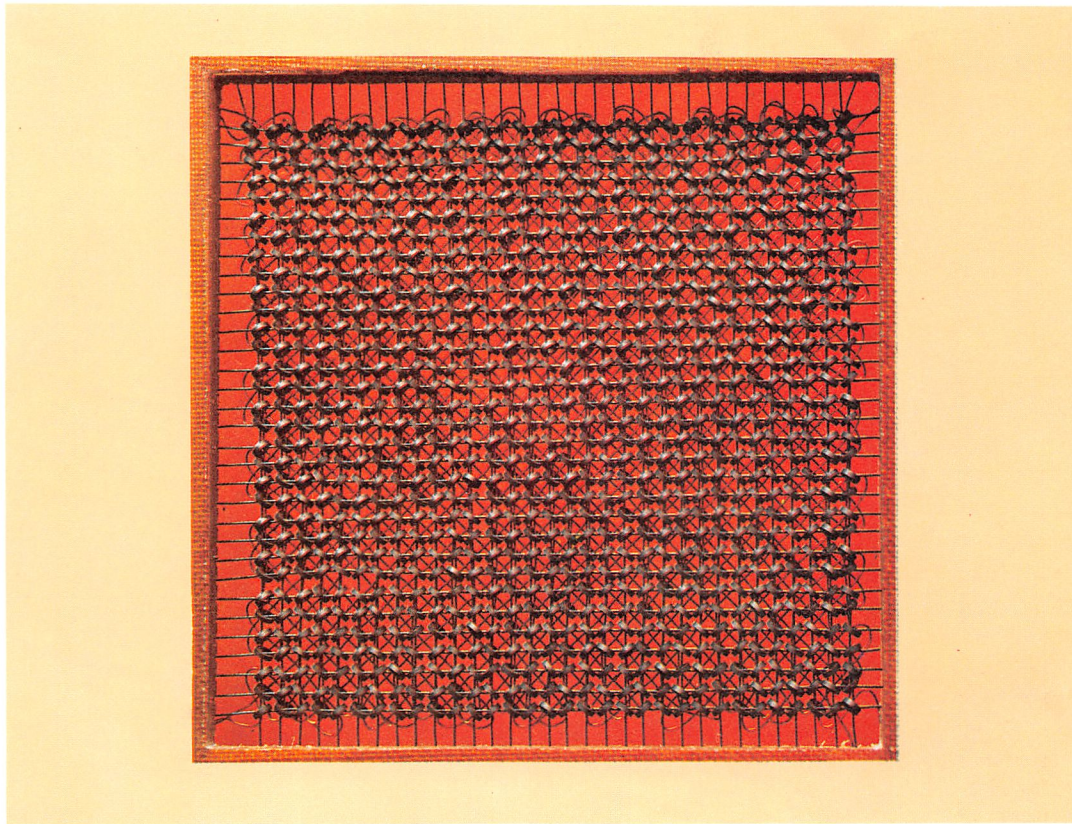
An example of non-volatile memory is the **magnetic core** memory shown in *figure 20*. This type of memory consists of many small ferromagnetic rings called

18. Memory organisation in a system with an eight-bit address. Six bits address each cell, leaving the two most significant bits to address the blocks.



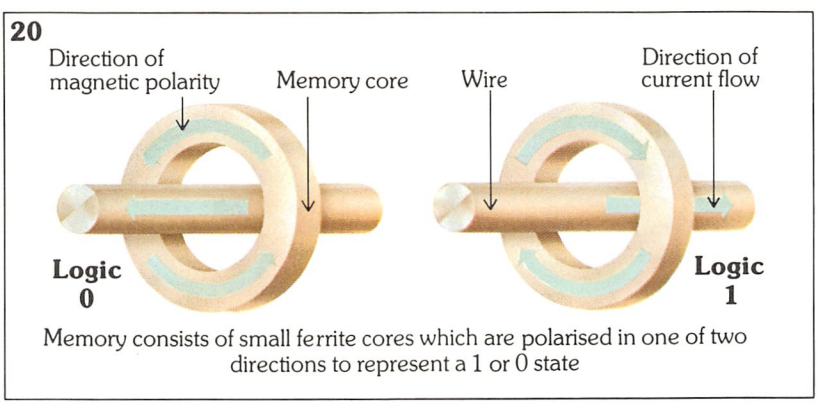


19. Cross coupling of two inverters: the typical circuit for a volatile semiconductor memory, called a latch or flip-flop.



ZEFA

Right: A matrix of ferromagnetic rings, known as cores, forming part of a magnetic core memory, each holding one bit. Each ring is about the size of a pinhead. Because of its relative disadvantages in terms of speed and size, core memory has now been superseded by semiconductor memory.



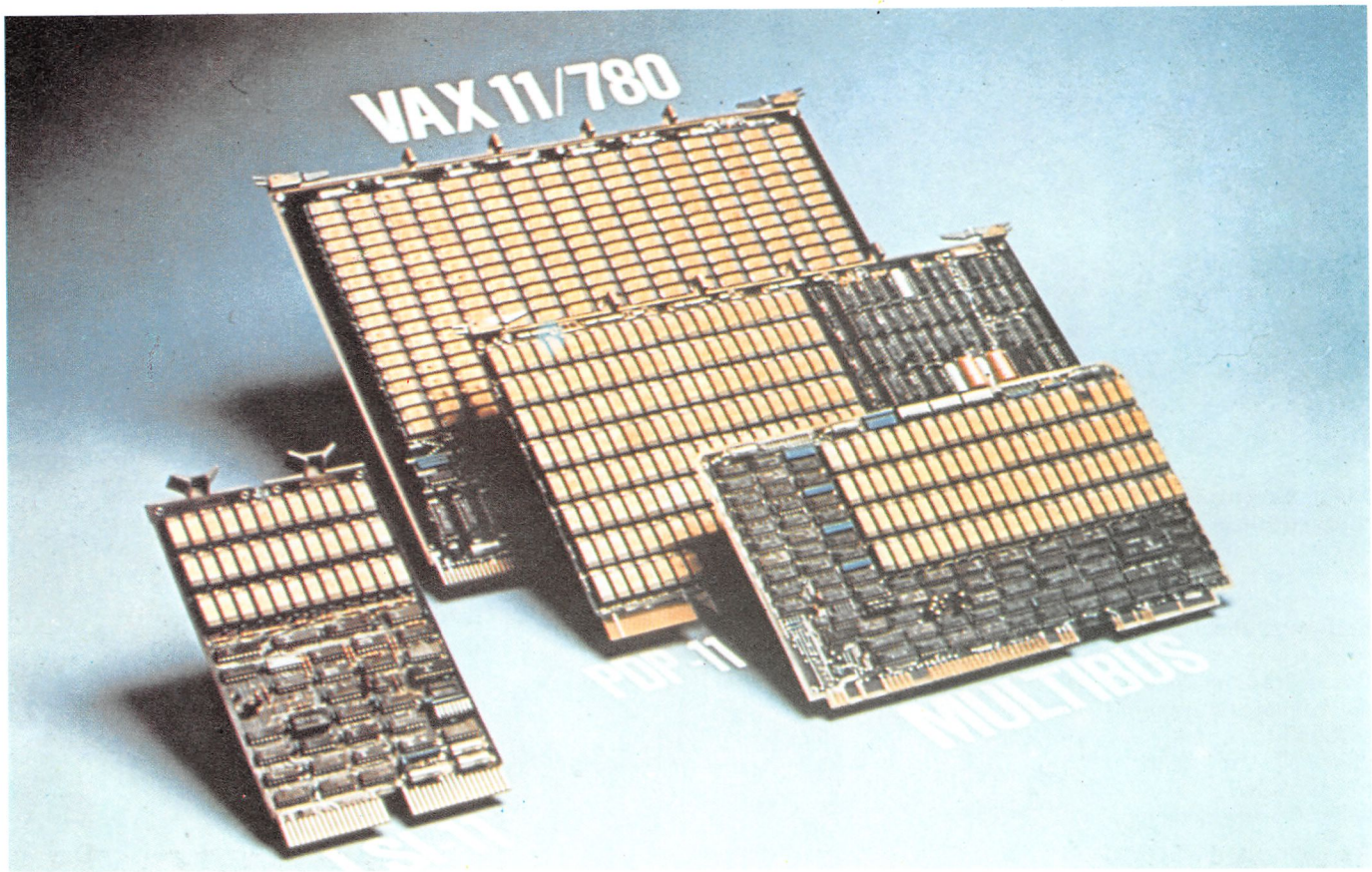
20. The two different states of a magnetic core memory.

cores. Each ring holds one bit. The ring is magnetized with one polarity for logic 0 and the other for logic 1. Core memory is relatively inexpensive, but the time needed to write data into or read data from the memory is much longer than for a semiconductor memory.

Core memory is also physically much bigger than a semiconductor memory of equivalent capacity. Even though each ferromagnetic ring is only the size of a pin-head, eight of them are required for each byte. This means that a 64K byte memory (i.e. 65,536 bytes) would need over half a million of these rings.

For these reasons, most modern computers no longer use core memories. Some of the later models of core memory based computers also used fast semiconductor memories. The faster access memories contain the most needed instructions and data. This speeds up access times. Such memories are called **cache** or **buffer** memories. As far as the user is concerned the data appears to go straight from the core memory to the CPU. But what in fact happens is that a certain block of data from the core memory is moved temporarily into the semiconductor memory, and all future accesses to this block are then made on the semiconductor memory. When an address outside this block is called for a new block of data is transferred from core to buffer.

There are other types of memory which, if not in widespread use today, may well be in the future. Of these, the **bubble memory** is worth a mention. This is a non-volatile memory which requires less power and less space than core memory. Physically it is about the size of an integrated circuit, and it uses tiny magnetised areas called bubbles in a specially



manufactured crystalline material.

Present day bubble memories are slower than semiconductor and core memories, but they are likely to have a lot of impact in replacing some auxiliary storage devices. Auxiliary storage devices are intrinsically slower than either semiconductor or core memories, but they can store more information at less cost than the main memory.

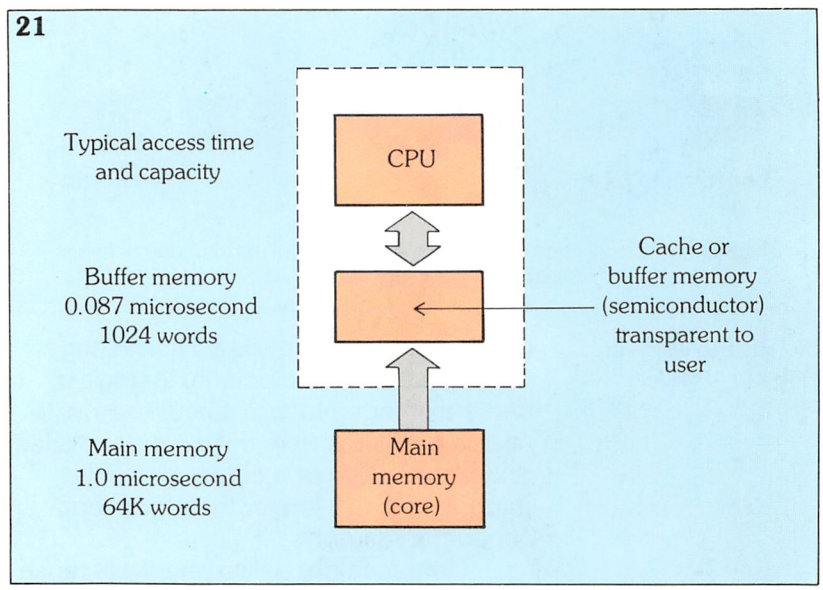
Mass Memory

Central memory is relatively expensive, so it is used only to store the computer's main program and related data. To store large amounts of data or additional programs, auxiliary or **mass storage** devices are used. The time taken to retrieve information from these is longer than that required by semiconductor or core memory.

Access time

The time needed to retrieve a particular word or byte is called the **access time.** This time starts from when the address is sent to the device, and ends when the device responds with the appropriate information.

In general, storage devices can be split into three groups depending on the way they are accessed. These are: random



21. Faster access memories are known as cache or buffer memories. These contain the most needed instructions and data.

Left: Examples of various types and sizes of bubble memory. These are non-volatile and require less space and less power than core memories.

access, direct access and sequential access. For **random access** the time needed to recall an item of data is fixed, regardless of its address. A typical example of a random access memory is the computer's central memory.

For **direct access** the time taken to recall an item of data depends on the location of the address, but access is direct because it is not necessary to search through a number of other data items to find the desired one.

For **sequential access** all data items occurring before the one addressed must be accessed before obtaining the desired one. A typical example of a sequential access device is a magnetic tape mass storage unit.

Input-Output operations

Looking back to *figure 3* we see input/output units connected to the CPU via an interface. The most common input/output units (in general they're called **peripherals**) are keyboards, printers, punched card readers, card punches, paper tape readers and punches, magnetic tapes, floppy disks, hard magnetic disks, magnetic drums and video displays. Every peripheral has its own hardware characteristics which must be matched to the characteristics of the CPU.

Every peripheral has its own unique address. This address is sent along the address bus when the CPU wants to access a particular device. Data transfers between the CPU and peripherals are carried out on the data bus.

There are two ways in which the CPU can contact peripherals. The CPU may be used to control all the transfers between itself and the peripheral. This function is called **program controlled transfer**.

Alternatively, the CPU simply specifies which device is to be accessed, and what sort of transfer is to be carried out. By the time the data transfer is finished the computer is doing something else. This rapid method of transfer is called **direct memory access** (usually shortened to **DMA**). A more detailed description of this is given in a later chapter.

Below: Each of these integrated circuits has a 64K Random Access Memory (RAM).

Glossary

adder	part of a basic computer circuit which provides a sum and a carry when adding two binary numbers
address	a pattern of characters that identifies a unique storage location
ALU	arithmetic and logic unit, a subsystem of the CPU that can perform arithmetical and logical processes
Boolean algebra	named after George Boole, who first used mathematical notation to express logical relationships in the same way that conventional algebra is used to express mathematical relationships
bus	two or more conductors running in parallel used for carrying information
direct access	type of memory where access time depends on address location, but it is not necessary to search through a number of data items to find a particular item
flag	an indicator which if in a status register gives information about the state of the CPU, or if attached to a data item gives information about the data item itself eg. will show if the data item has caused an error
flip-flop	a logic circuit which stores one bit, the logical status of which is changed by an input signal – it flips and flops from 1 to 0 and back
interface	the control circuits that link the CPU and its peripherals
logic	in computers, the systematic scheme used to define how information in binary form is manipulated
magnetic core memory	memory consisting of tiny ferromagnetic rings called cores. Each ring, which holds one bit, is magnetized with one polarity for logic 0 and the other for logic 1
multiprocessing	the process of running two or more programs simultaneously in a computer which has more than one independent ALU in its CPU
random access	random access memory (RAM) is the type normally used in a computer's central memory. The time needed to access any item is the same, regardless of its address
sequential access	in this type of memory, all data items have to be accessed in sequence before the desired one is obtained
truth table	table derived from Boolean logic showing the logic state of each output that results from each combination of logic states (0 or 1) at its inputs
volatile memory	a memory made of semiconductors which loses its contents when switched off; non-volatile memories do not lose their contents when switched off

Semiconductors and how they are used

Semiconductors and systems

While all electronic systems share basic similarities – that is, they can be broken down into the sense, decide and act stages and they all either manipulate information or do work – they differ enormously within this framework, as we have already seen. Obviously a circuit for a radio would be very different from that needed by a watch. The semiconductors they use also vary greatly and are designed and chosen according to the type of work they are required to do.

Even within the same system the various components have different functions and need to employ a variety of semiconductors. So before you can begin to understand the thinking behind a particular system, you'll need to know what types of semiconductor are available.

Although a semiconductor is, strictly speaking, a piece of silicon or germanium used in a device, the devices themselves are commonly referred to as 'semiconductors', as is the case in this chapter.

What are the broad categories of semiconductor devices?

It is useful to group semiconductors into a few broad categories to help understand the various uses they can be put to. Within these categories there are other differences which lead a designer to choose one rather than another, but these do not concern us at the moment.

Figure 1 shows the 'family tree' of semiconductor devices which is divided into categories according to their suitability for different applications. Starting at the top of the tree, we can see that the family of semiconductors can be divided into **interface** or **all-electrical** devices.

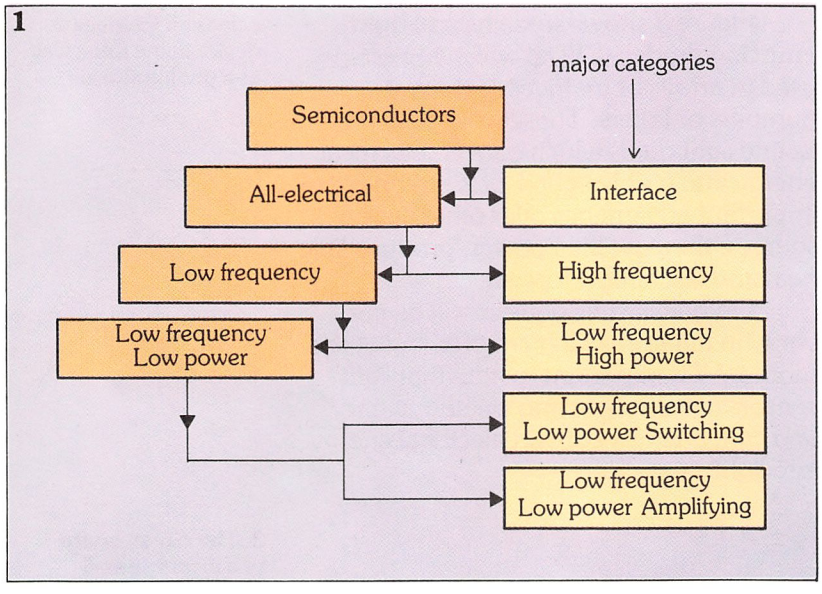
Interface devices are the only types of semiconductors which directly interact with the outside world in that they can handle non-electrical input and output. This category is rather small and is chiefly made up of optoelectronic devices such as light sensors or light emitters (LEDs).

All-electrical semiconductors, as the name implies, work only with other electrical or electronic components and circuitry.

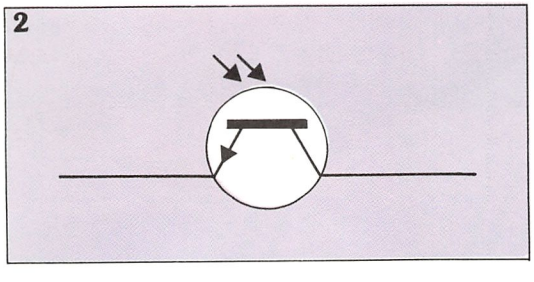
Both these types of device are divided into two groups, low frequency and high frequency, which determines how they can be applied. Frequency is the rate at which a signal wave changes, measured in the number of cycles of change that occur in one second (hertz).

All semiconductor devices can be used for switching or amplification, whether low or high frequency types.

1. The semiconductor family tree, showing the way that semiconductor devices are grouped according to their suitability for different applications.



2. The usual symbol for a phototransistor. This is an example of an interface device.



Some however are designed to perform switching more efficiently and others to perform amplification better. High frequency devices can usually act as faster switches than low frequency ones. Both high and low frequency devices can be further divided according to their power-handling capabilities.

The column at the right hand side of the tree shows which groups of semiconductors are suitable for particular uses. Let's move on to look at some of these applications.

Interface devices

Interface devices either sense external energy or produce it. That is they act as an intermediary between the outside world and the inner workings of the electronic system.

The heat sensitive resistor (**thermistor**) is a typical interface device. It may not in fact be a semiconductor but offers a good example of how an interface device functions. The resistance of the thermistor varies with temperature and so it can sense external energy in the form of heat and transmit this information electrically.

Optoelectronic devices are true semiconductors. *Figure 2* shows the electrical symbol for a **phototransistor,** such as you might find used in a street light control system. You will notice that it has an emitter and a collector just like an ordinary transistor, but that it has no electrical connection to the base region of the semiconductor element. This is because energy contained in the external light passes through a little window or lens in the device enclosure and generates what amounts to a base current which turns on the transistor.

Figure 3 shows some typical **light emitting diodes** (LEDs) which are often used in arrays or matrices to display numbers or letters. These work very much as tiny light bulbs which convert electrical energy into visible light. LEDs have several important advantages over other light sources: they use little power, produce little heat and last almost forever.

Optoelectronic devices will be covered in detail in a later chapter, but at the moment it is important to note that *light sensors* convert light into electric signals and *light emitters* convert electrical energy into light.

3

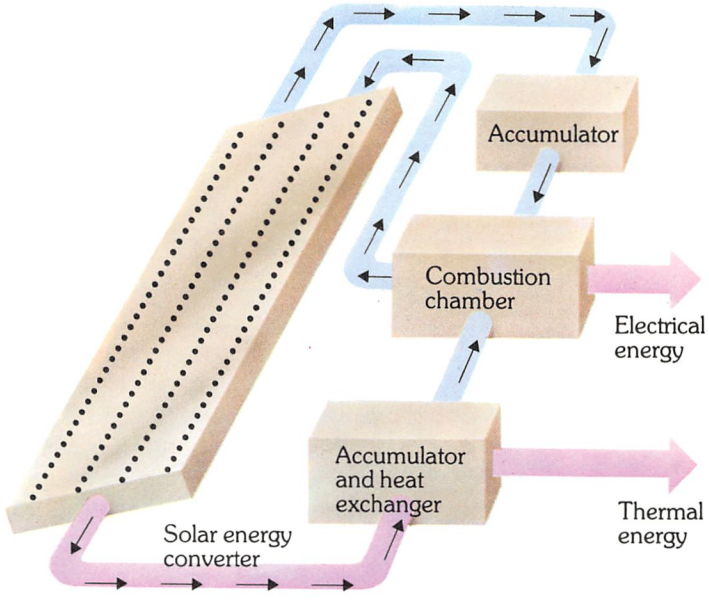
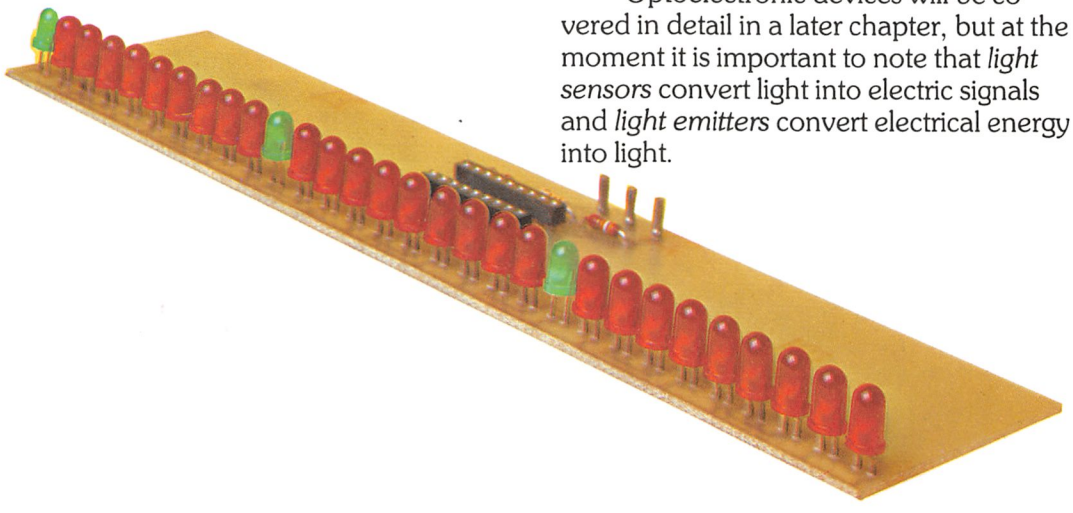


Diagram of a solar powered generator, where the sun's energy is transformed into electricity and directly usable heat. Essentially this system acts as an interface device in the same way as a phototransistor.



3. The circuit board of a display panel using light emitting diodes as indicators.

How frequency affects circuits and semiconductors

The division of high frequency and low frequency is, of course, an oversimplification and there is no sharp dividing line as such. Generally speaking, the performance of a typical low frequency circuit will noticeably deteriorate at around 300 kilohertz (300,000 cycles per second) and will probably be completely useless at around 300 megahertz (3 hundred million cycles per second). However, some low power circuits can operate satisfactorily up to three gigahertz (3×10^9 cycles per second) or even higher.

Where are you likely to come across high frequencies? High frequencies are useful both in digital and analogue equipment. In digital computers the high frequencies of the switch circuits, which currently go up to 200 megahertz, make possible the incredibly fast operation and the extraordinary 'number-crunching' capacity of modern computers. Even higher frequencies are needed in some analogue equipment using amplifying circuits; in telecommunications, radio waves are received and transmitted over a very wide frequency spectrum which can reach hundreds of gigahertz (billions of cycles per second).

Very low frequency radio waves – from 10 to 30 kilohertz – are used in transmissions to submerged submarines and low frequency waves are also used by surface ships. These transmissions can travel great distances because they are reflected by the ionosphere and bounced back to earth. Very high frequency waves cannot do this.

The various broadcasting systems use a range of frequencies. Our familiar radio set comes into the medium frequency range which is used for AM broadcasting and some aircraft purposes. High frequencies are used for short-wave broadcasting over long distance while the very high ones (VHF) are used for FM radio broadcasting and television. Even higher than these are the micro-wave frequencies needed for line-of-sight radio communication and radar. Beyond these frequencies we get infra-red light and the visible spectrum.

How power affects circuits and semiconductors

Before discussing the pros and cons of power in relation to semiconductors, we need to explain the concept of dissipation.

One of the main characteristics of semiconductors – and many other electrical devices – is the amount of power they can dissipate. In simple terms power dissipation means the heat which is generated within a device by the friction of the electrons which are flowing through it.

A good way to illustrate dissipation is to compare the device to your hand and the current to a rope being pulled through your grasp. As the rope is pulled your hand will be heated by friction – a certain amount of the power applied to the moving rope is being dissipated (wasted) in the form of heat.

If heat is generated too quickly within a device (i.e. faster than it can be taken away by the chassis or surrounding air), it will get hotter and hotter. Excessive temperatures may cause malfunctioning of the device and it may even burn out. For this reason devices have a **power dissipation rating** which indicates the rate at which heat can be generated within them without doing any damage. This rate of heat generation is measured in watts or milliwatts ($= 10^{-3}$), which are units of power.

High and low power semiconductors

In terms of this rating there is no sharp dividing line between high power and low power devices: although some semiconductors can only take a few milliwatts, others take several watts and some are made to dissipate hundreds of watts.

It is important to realize that the power dissipated by a device is not equal to the electrical power transmitted through it. But there is a distinct relationship between these two quantities.

Think once again of your hand grasping a moving rope. The heat is governed by two factors – the speed of the rope (which is equal to the electric current) and the force which your grip exerts on it (equal to the voltage pressure exerted on the device: i.e. to the potential difference

between one side and the other). The faster the rope moves and the tighter your grip, the hotter your hand gets. In the same way, the greater the current and the bigger the voltage drop, the greater is the power dissipation.

To illustrate the importance of power dissipation *figure 4* shows a simple motor control circuit. There is a 12 V power supply, an NPN amplifying transistor of power rating 1.8 W, a motor and an earth. Imagine that only enough electrons are being taken from the base of the transistor to allow a current of one amp to flow from the emitter to the collector. Suppose, also, that this is enough to maintain a voltage of 2 V on the wire to the motor.

In this example the power dissipated in the transistor means the power wasted in the process of regulation. Using the standard formula amps x volts = watts, the power dissipated in the transistor (wattage) can be calculated by multiplying the drop in voltage across the transistor by the current flowing in the circuit. As there is 12 V on one side of the transistor and 2 V on the other side, the drop in voltage is 10 V. Therefore, the power dissipated is:

$1\,A \times 10\,V = 10\,W.$

But the power dissipation rating of the transistor is only 1.8 watts, so what happens when 10 watts are generated? All this power must flow through a tiny silicon chip and the heat is generated within it. The smaller the chip is, the hotter it will get. When the temperature of silicon goes beyond 200°C, the transistor goes out of control and will soon be destroyed. In short this transistor is unsuitable for use here.

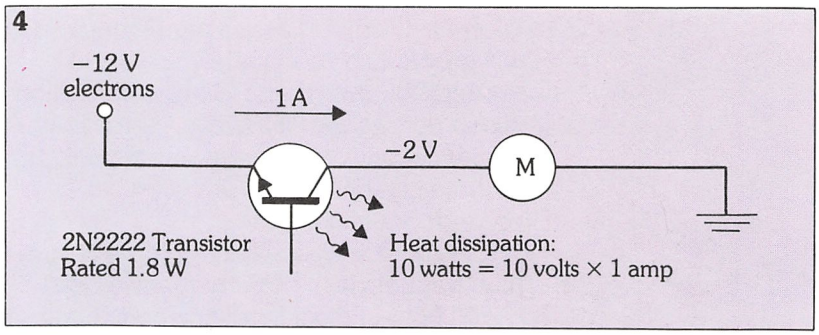
Why are high and low power semiconductors different?

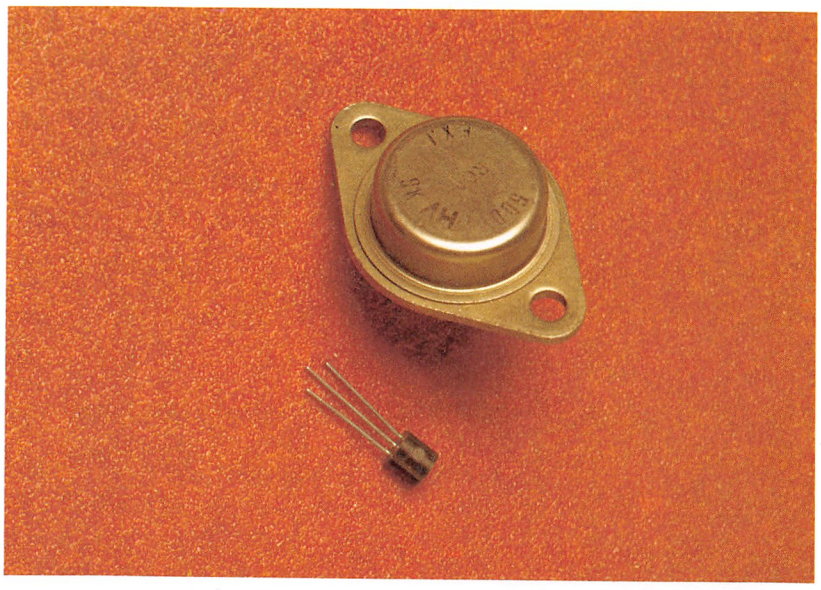
So what kind of transistor is needed for the system to function properly? A bigger semiconductor chip is necessary – this will concentrate the heat less and keep the temperature lower. Better contact between the chip and its case is needed – this will cut down heat insulation and allow heat to flow out of the chip more easily. Finally a bigger case is needed – to give a greater area for heat transference to the surrounding air or to the heat conduction plate of the device. The suitable high power rated device will have a big chip, a bigger case and thicker leads to take more current.

Figure 5 shows a high power device which would solve the problem. Transistors capable of handling high power are known simply as **power transistors.** A small signal low power device is beside it for comparison. The differences are obvious. This power transistor can dissipate up to 50 watts – leaving a more than generous safety margin in our circuit. Note the bolt holes which enable the device to be tightly mounted to the chassis or to a large **heat sink**. These are a common feature of power transistors. Other methods of cooling include special air blowers or even water circulation.

You may well ask, 'Why not make all transistors capable of handling high power?' Then there would be no need to worry about malfunctioning caused by heat. In the first place they would be too bulky. A system as small as a cigarette packet can contain hundreds of low power transistors.

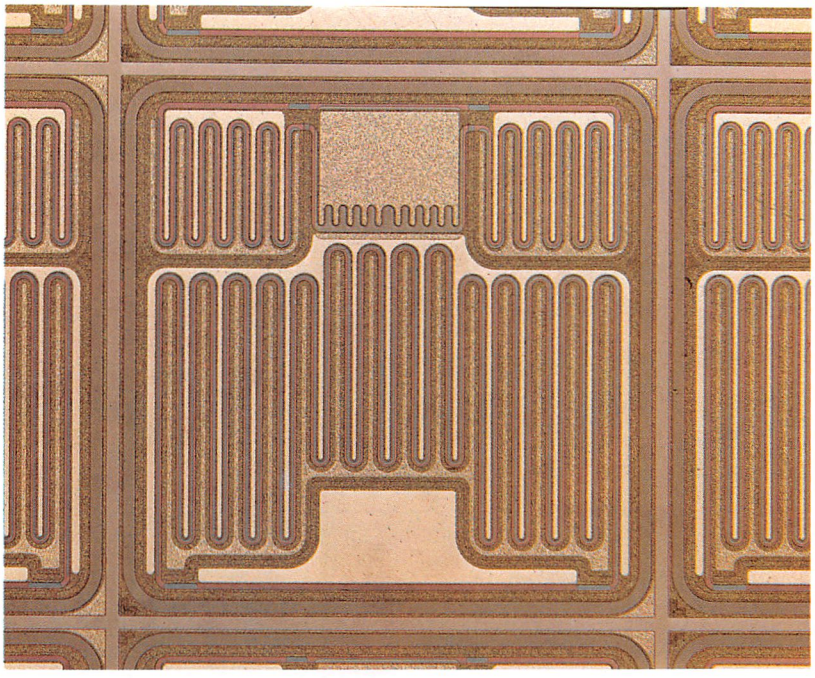
The circuit board of a 'sustain' pedal for an electric guitar. It shows transistors, capacitors, resistors and connecting tags.





4. A simple switching circuit for a motor, indicating the heat dissipated in the transistor.

5. High and low power transistors shown actual size. (The larger of the two is the high power device).



If these were substituted by power transistors you would end up with a system as big as a shoe box. Heat is another consideration; this system would generate as much heat as an electric iron and would need a special cooling system. Worse still, it would switch at very low speeds, so it would be impractical for computer applications. It would also be prohibitively expensive. As you can see, there are definite reasons for setting aside power semiconductors as a group.

Power semiconductors are distinguished not only by their appearance and power rating but also by their applications. They are principally found in the 'act' stage of systems where they switch and regulate the power controlling the working devices. As well as transistors, other types of semi-conductors such as diodes and thyristors are made in high and low power ratings.

Glossary

all-electrical semiconductor	these work only with other electrical or electronic components and circuitry unlike interface devices which interact with the outside world
heat sink	large piece of metal which acts as a heat radiator, to which the device is bolted
interface device	semiconductor devices which directly interact with the outside world (see all-electrical semiconductors)
power dissipation	whenever electric current flows from a higher to a lower voltage – such as through a motor, resistor or transistor – a certain amount of energy has to come out of it (measured in watts). If this energy or power is not converted to work (for example, by a motor), it is dissipated – that is, wasted in the form of heat
thermistor	an interface device, the resistance of which changes proportionally with temperature
watts	the units in which power is measured. Watts are calculated by multiplying the current by the voltage: amps x volts = watts

BASIC THEORY REFRESHER

ELECTRICAL TECHNOLOGY

Amps, volts and ohms

An electric current is the ordered movement of electrons through a conductor. Everyone knows that a bulb lights up when it is connected to the terminals of a battery. The current flowing through the bulb from the battery is what causes it to light up. The more electrons that flow through the bulb, the higher the current.

Even with a very low current the number of electrons passing along the conductor in one second will be enormous. As a unit of measurement of current (symbol I) we use the ampere (**amp**), or its multiples or submultiples. One amp represents 6,240,000,000,000,000,000 (6.24×10^{18}) electrons passing through the conductor in one second.

Similarly:

1 kiloamp = 1 kA = 1000 A = 1×10^3 A
1 milliamp = 1 mA = $\frac{1}{1000}$ A = 1×10^{-3} A
1 microamp = 1 μA = $\frac{1}{1000000}$ A = 1×10^{-6} A

An instrument called an **ammeter** is used to measure the current.

But what causes the current to flow? Since the current is the movement of free electrons, it's obvious that some force must be causing the movement. The name given to this force is **electromotive force** or **EMF.** Every source of electrical energy is an EMF generator. Various sources of EMF are available; batteries for example are chemical sources, while the dynamo on a bicycle or car and the massive generators that supply our homes are mechanical sources. Nowadays solar cells are becoming popular; they convert the sun's heat energy directly into electricity.

All of these sources have a terminal which supplies electrons and another which receives them. On a battery or dynamo these are respectively called the **negative** (−) and **positive** (+) **terminals.**

The negative terminal has an excess of free electrons, while the positive one has a lack of them. We say that one has a negative potential and the other a positive potential and call the difference the **potential difference.** The purpose of the EMF generator is to create and maintain the potential difference between the two terminals. When a conductor is connected to the two terminals they try to equalise their potentials by passing electrons through the conductor from the negative to the positive, as in *figure 1*.

The source of EMF, in order to maintain the potential difference between the terminals and thus maintain the flow of electrons, has to supply continuously a form of electrical pressure rather like water pressure is needed to keep water flowing through a garden hose. Obviously for a current to flow there must be a potential difference. The potential difference between the terminals of an EMF source is measured in volts (symbol V) and an instrument called a **voltmeter** us used to measure it.

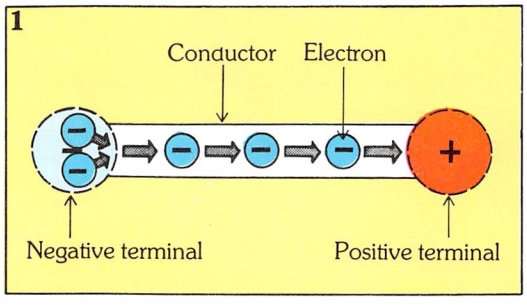
1 kilovolt = 1 kV = 1000 V = 1×10^3 V

1 millivolt = 1 mV = $\frac{1}{1000}$ V = 1×10^{-3} V

1 microvolt = 1 μV = $\frac{1}{1000000}$ V = 1×10^{-6} V

To work properly any electrical device must be connected to the correct voltage

1



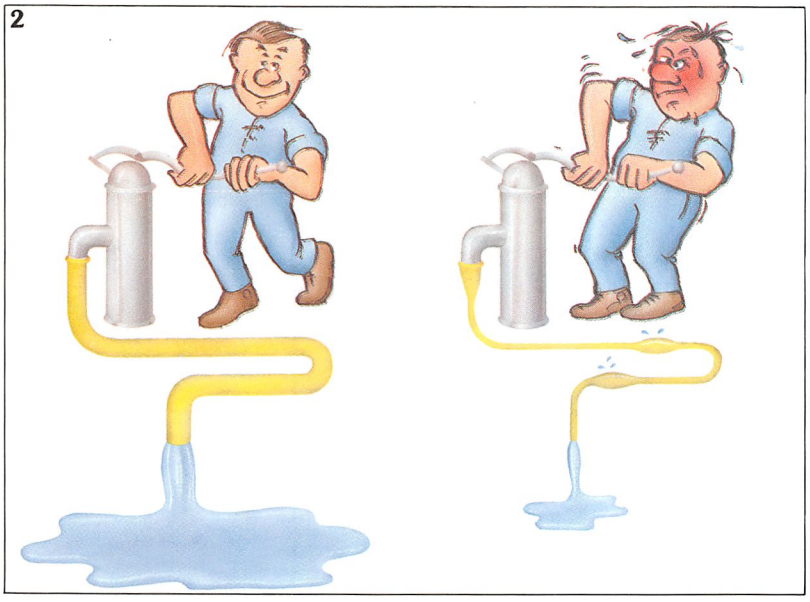
1. Negatively charged electrons at the negative terminal will move down a conductor to the positive terminal, where the negative and positive charges will cancel each other.

Table 1

Resistivity of the major conductive materials

Material	Resistivity (Ω m)	Temperature coefficient (at 20°C)
Silver 99.99% pure	0.016	3.8
Electrolytic copper	0.0177	3.9
Gold	0.023	3.4 – 3.8
Crude aluminium – 99.9% pure	0.028	4
Tungsten	0.055	4.5
Phosphorous bronze (telephone lines)	0.07	3.9
Pure nickel	0.072	6
Steel (wire)	0.1 – 0.25	4.5 – 5
Iron 99% pure	0.1 – 0.15	4.5
Platinum	0.1	3.6
Lead	0.21	4
Argentan 60Cu/25Zn/15Ni	0.35 – 0.4	0.07
Manganin 80Cu/15Mn/5Ni	0.42 – 0.46	0.01
Costantan 60Cu/40Ni	0.5 – 0.51	circa 0
Cast iron	0.6 – 1.5	—
Nichrome 80Ni/20Cr	0.9 – 1.1	0.11 – 0.16
Mercury	0.95	0.89
Graphite	4 – 20	—
Carbon, amorphous	38 – 40	—
Brush carbon	20 – 100	—

NB. Resistivity figures are expressed as $10^6 \times$ their true value. Temperature co-efficient figures are expressed as $10^3 \times$ their true value.



2. A short wide pipe allows water to pass through it more easily than a long thin one. So too in electricity resistance is affected by the length of the conductive material and its cross-sectional area.

Battery	⊣⊢	Generator	G
Ammeter	A	Voltmeter	V
Conductor	—	Lamp	⊗
Resistance	▭		

Graphic symbols normally used in schematic diagrams to represent various electrical elements.

source. Too high a voltage could well destroy the device, too low and it may function poorly or not at all.

Another important factor that controls the flow of electricity is **electrical resistance.** The analogy of the garden hose can help explain it.

It's fairly easy to see that the rate at which the water flows through the pipe depends (in addition to the pressure at the tap) on the resistance it meets going through the pipe. And the longer the pipe, the more resistance there will be. On the other hand if a large diameter pipe is used it will have less resistance and the water will flow more easily *(figure 2)*. Also,the water will flow more easily through a smooth pipe than one with a rough surface.

So, for any given pressure the flow of water depends on the length of the pipe, the diameter of the pipe, and the type of the pipe. Similarly, for an equal EMF, the amount of current flowing in a conductor depends on the length of the conductor, its cross sectional area and the type of material used.

These three things all contribute to the resistance of a conductor, which is a physical property and doesn't depend on the current flowing through it. This means that in a circuit the current and voltage can be varied but the resistance effectively stays the same.

Electrical resistance (symbol R) is measured in **ohms** (symbol Ω). One ohm is defined as the resistance which allows one amp to flow when a voltage of one volt is applied across it.

1 kilohm = 1 kΩ = 1000 Ω = 1×10^3 Ω

1 milliohm = 1 mΩ = $1/1000$ Ω = 1×10^{-3} Ω

1 megohm = 1 MΩ = 1000000 Ω = 1×10^6 Ω

So for example a piece of copper wire may have a resistance of 1A. If an EMF of 1 V was placed across its ends then a current of 1 A would flow along the wire. An exactly similar piece of wire in length and thickness made of aluminium would however have a resistance of about 1.6 Ω and a proportionately smaller current would flow through it when an EMF of 1 V was placed across its ends.

All materials have different resistivities (symbol ρ). Resistivity is defined as the resistance of a wire made of the material which is 1 metre long and 1 square metre in cross-sectional area, measured in Ω m. If we know the dimensions of a conductor and its material we can calculate its resistance as follows:

$$\text{resistivity} = \text{resistance} \times \frac{\text{cross sectional area}}{\text{length}}$$

Table 1 shows both the resistivity of various materials and the effect on the resistivity of changing their temperature. The **temperature co-efficient** of resistivity of a material is the fractional increase in its resistivity for each 1° centigrade rise in temperature.

In some calculations it is useful to use values of **conductance** (symbol G) rather than resistance. This represents the ease with which a material allows electricity to pass through it. The conductance is therefore the inverse of the resistance.

$$G = \frac{1}{R}$$

Conductance is measured in units known as **Siemens** (symbol S). □

BASIC THEORY REFRESHER

ELECTRICAL TECHNOLOGY

Elements of a circuit

When electricity was first discovered it was assumed that current flowed from a positive terminal, through a circuit, to a negative terminal. All the conventions about how electricity flowed were based on this.

However, as science progressed, it was found that in reality the current flow was from negative to positive. In order to allow the conventions defined up to then to remain true, it was necessary to differentiate between **real** current flow and what was, up to then, the supposed current flow. This was done by calling the supposed current flow **conventional current** and the real current flow **electron current flow**.

Of course, no current will flow until a path is made between the two terminals of a power supply, creating what is known as an **electric circuit**. An electric circuit always consists of the following elements:

– a voltage supply (generator, battery, etc)
– a load or work element, i.e. the object that is receiving the supply
– connecting lines (conductors)
– normally a switch is also included.

Any voltage source can be considered as a pure generator of EMF (E) plus a resistance (R) — see *figure 2a*. When a current flows through the resistance, some of the EMF is used up in 'pushing' the current through it. Because of this the voltage felt at the terminals of a supply is only ever equal to the EMF when no current is flowing. As soon as current starts to flow, the voltage drops. This is expressed as

$$V = E - V_d$$

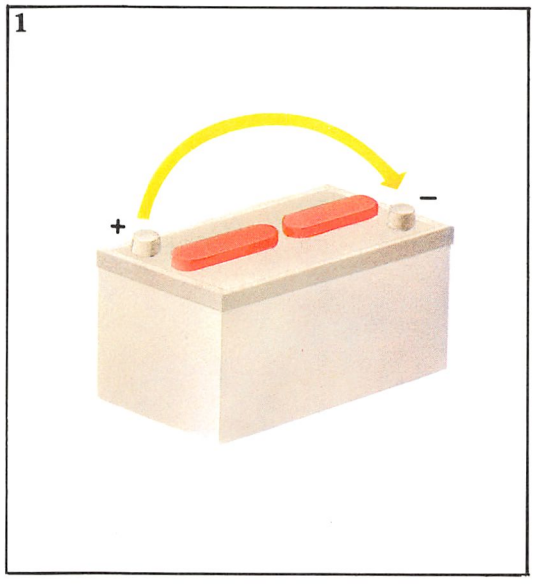
where V_d is the amount of electromotive force lost in pushing the current through the resistance. If the supply has to provide a lot of current, more of the EMF is used to push the current through the internal resistance, and so the voltage on the terminals drops even more.

The load in an electrical circuit is anything that is doing a job. A television for example, is a load, and so is a lamp. Obviously, if you had to know the workings of, say, a television in order to understand the basics of an electrical circuit, it would become a very involved business.

Fortunately this isn't necessary. A circuit can simply be regarded as a device with two external connections that will have certain electrical characteristics when measurements are made at those connections.

The simplest load is in fact a number of resistors connected together in some undefined way. It isn't necessary to know how many resistors there are, simply that they form a circuit with two terminals and a certain resistance. A single resistor, with the same equivalent value as the load can then be used to represent all the effects of the load on the circuit, as in *figure 2b*.

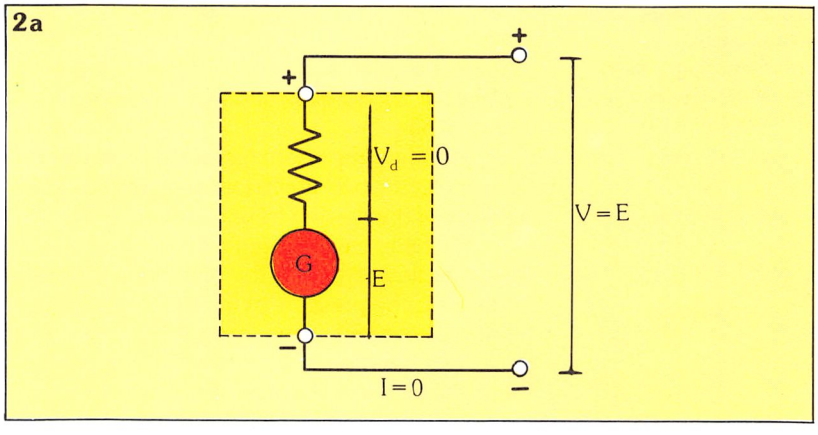
This sort of rationalization is a useful way of making it easier to understand the electric circuit laws.

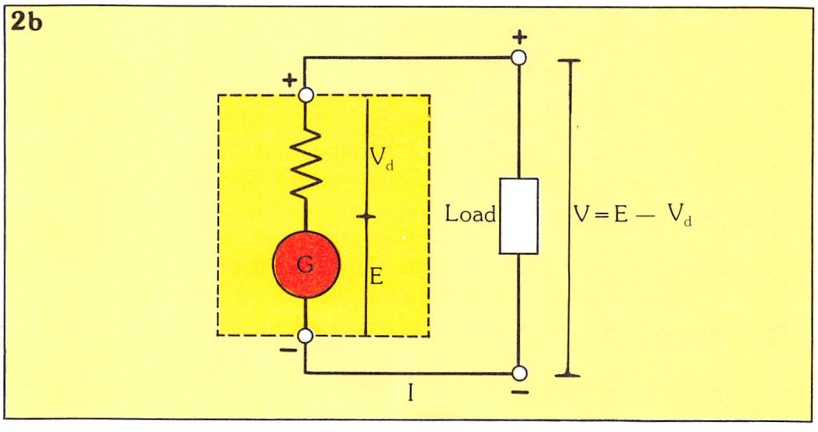


1. Conventional current is assumed to flow in the opposite direction to electron current: from positive to negative.

2 a and b. The effect of internal resistance on a source of power is to reduce the voltage across the terminals when current is flowing.

4. Colour coding chart used to identify the resistance value of carbon compound resistors.





3. The positioning of the coloured bands on a resistor.

So a power supply (such as a battery or dynamo) can be defined as a circuit element from which current flows out of the positive terminal. A load, on the other hand, is a circuit element in which current flows into the positive terminal. Since there is always this associated transfer of energy from the power supply to the load, the circuit elements can be defined in another way:

– power supply is an element that takes non-electrical energy (mechanical, thermal, chemical etc) and transforms it into electrical energy to be sent to the line (wire)

– a line is a circuit element that accepts energy from the generator, carries the energy along its length and then transfers it to the load

– a load is an element that takes electrical energy from the line and transforms it into other forms of energy (mechanical, chemical, thermal, etc depending on the function of the circuit).

In this way some types of battery can play a dual role. It can act as a source – supplying electrical energy at the cost of chemical energy – and when it is being charged it can act as a load, accepting electrical energy and converting it into chemical energy.

Since it should be possible to turn the electrical circuit off and on it is normal to add a switch, which is placed in the line that connects the generator to the load.

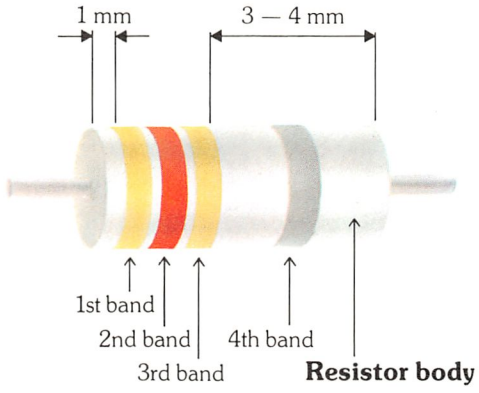
In some circuits it is necessary to insert resistors of a known value to achieve certain conditions. Obviously to have resistors of every possible value would be impractical, so resistors with standard values that effectively cover the whole range are produced.

Radio circuits, for example, make wide use of carbon compound resistors. On these the resistance value is indicated by a colour code. Colour coding is used so that the resistance can be easily identified while the component is in place in a circuit. The code consists of several coloured bands (see *figure 3*). The first band indicates the most significant number (in this case the yellow band indicates the figure 4 in 470,000 ohms). The second band indicates the second most significant number (e.g. the 7 in 470,000) while the third band indicates how many zeros have to be added to the first two numbers (e.g. 4 zeros in 470,000). This third band is in fact called the multiplier since it multiplies the first two numbers by the power of 10 (47×10^4 in the case of 470,000).

A fourth band is often included to indicate the tolerance of the resistor, that is to say the resistor could have an actual value that lies anywhere between plus and minus a certain percentage of the given value.

When the fourth band is missing, this automatically indicates a tolerance of + or – 20%. If you check the value of the coloured bands in *figure 3* against the *figure 4* chart, you will see that the resistor has a value of 470,000 ohms + or – 10%. □

3



4

Carbon resistors

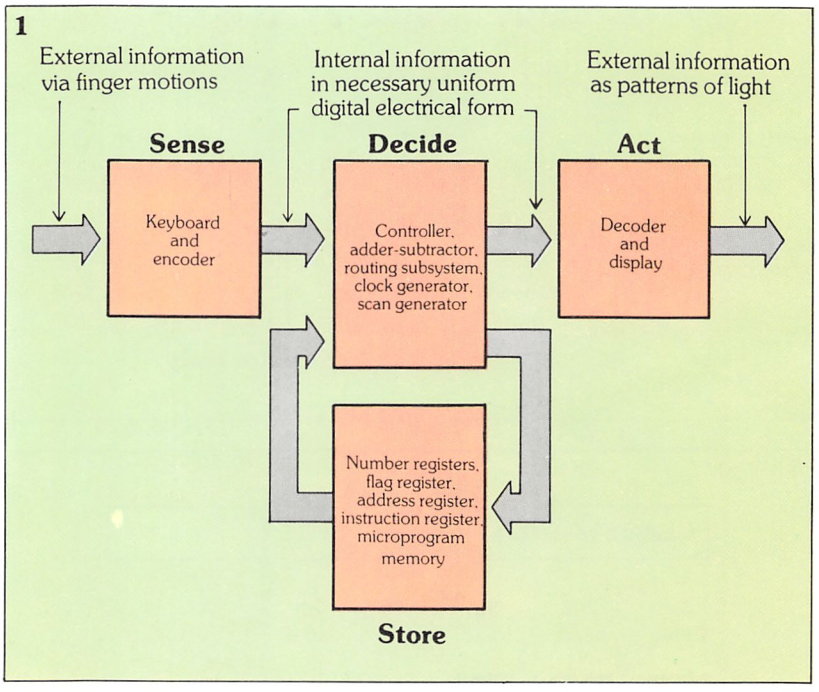
1st band	2nd band	3rd band	4th band
0	0	0	
1	1	0	
2	2	00	±2%
3	3	000	
4	4	0 000	
5	5	00 000	
6	6	000 000	
7	7		
8	8	0.1	gold ±5%
9	9	0.01	±10% silver

How digital circuits make decisions

Decision-making units

From chapter 1 you will now have a fair idea of how the switches in the calculator keyboard *sense* external information from your fingertips, and of how the LED display *acts* to produce external information in the form of patterns of light. We have also explained at a simple level how switching circuits can *store* information.

What is less clear at this stage is the *decide* function. How can electrical circuits make decisions? Do they have some sort of intelligence? It's a natural question to ask, but the answer is, of course, no. Understanding this process in more depth is our next step in studying how digital circuitry actually works.

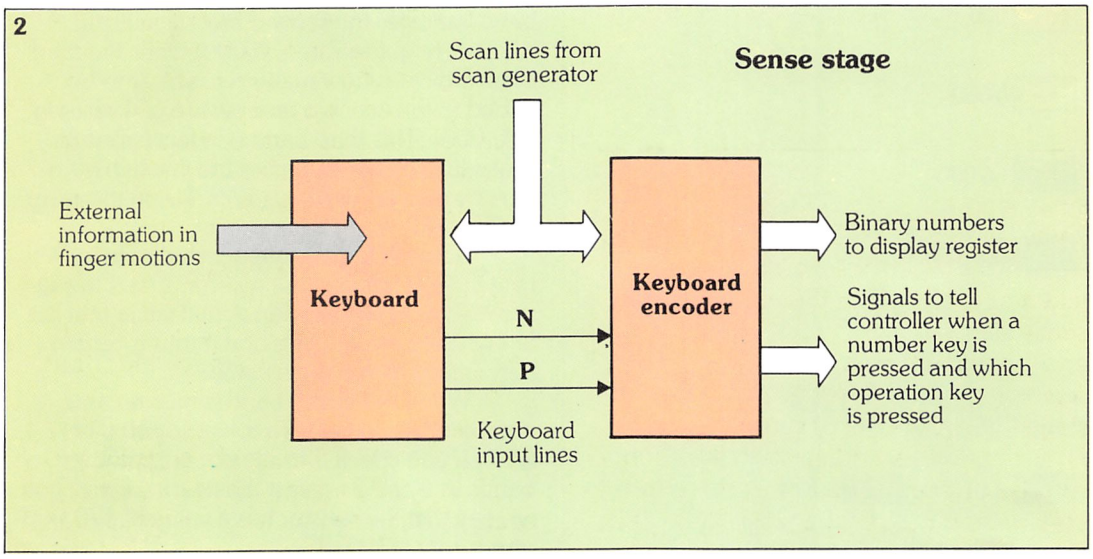


1. The organization of the calculator's subsystems showing the major functions they perform. A certain amount of decision-making is involved in all stages, but decisions are the *main* job only in the 'decide' subsystems.

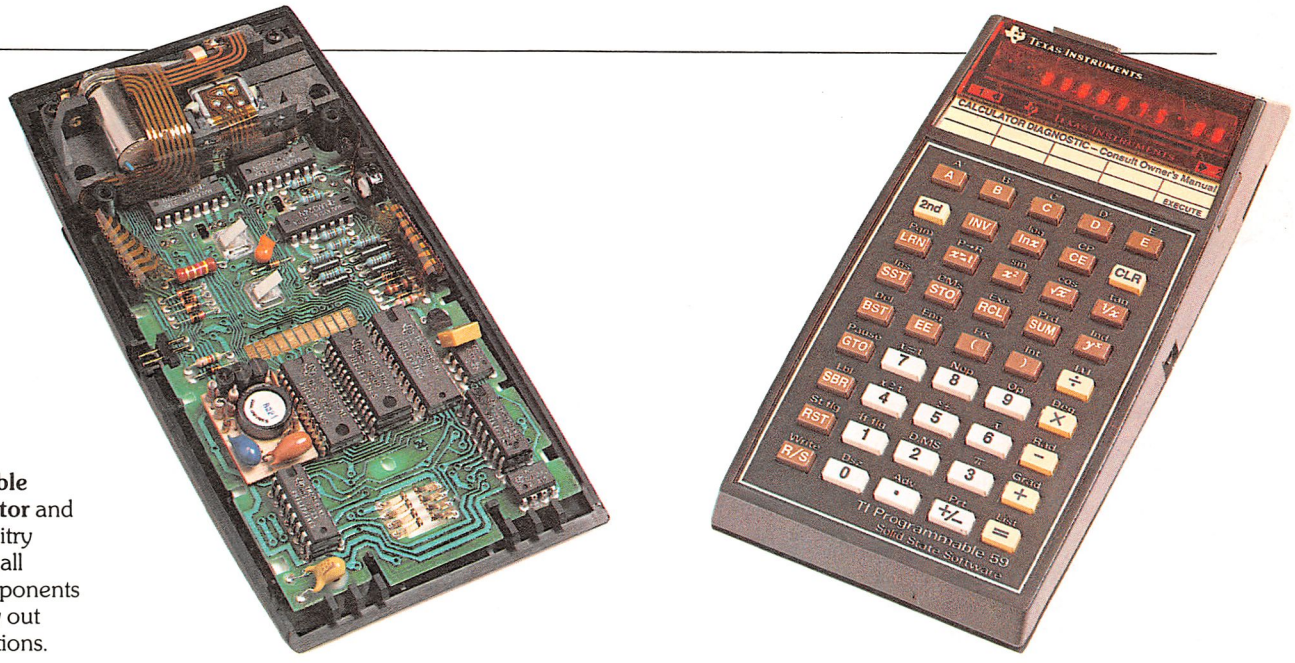
The simplest decisions in the calculator

Figure 1 shows how the different parts of the digital calculator's subsystems can be categorised according to which function they perform – whether their main job is to sense, decide, store or act. In reality a certain amount of decision-making is involved in each of the four stages, but decisions are the *main* job only in the area marked as the 'decide' unit.

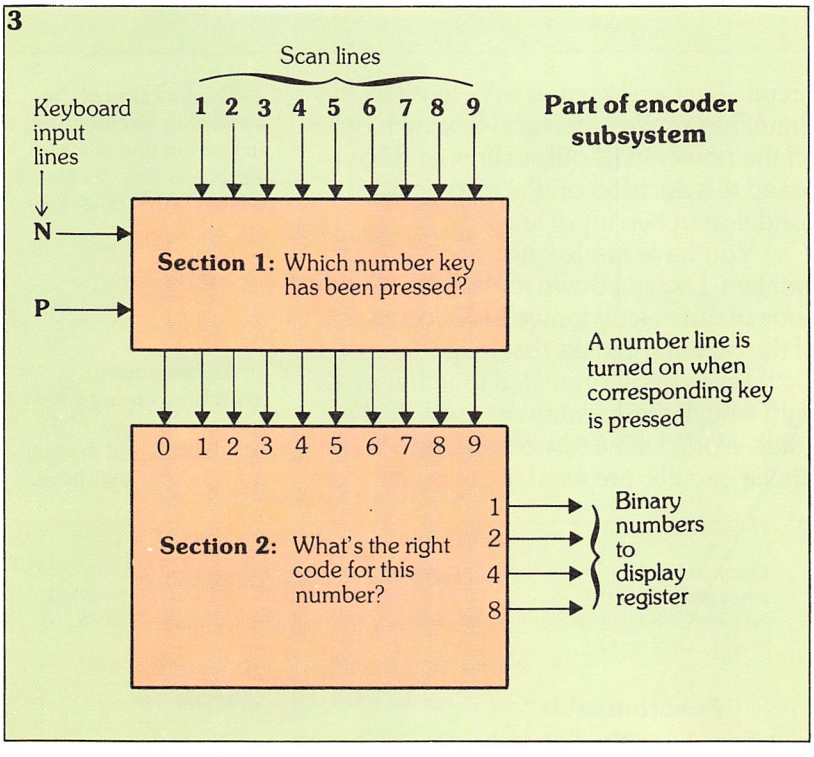
To grasp the basic idea, let's choose a very simple decision-making unit to study first. The most complex decisions are made in



2. The keyboard encoder senses incoming information and converts it to forms suitable for other subsystems. This conversion involves simple decision-making.



A programmable pocket calculator and its internal circuitry showing the small number of components needed to carry out complex operations.



3. The encoder takes two steps in converting keyboard signals into binary numbers.

the decide unit, so we'll start elsewhere and come back to this more complex area later.

The simplest decisions are in fact taken by the **keyboard encoder,** in the sense stage. *Figure 2* illustrates the encoder's job and why it is classified as part of the sense, rather than the decide, stage. This stage involves not only sensing external information supplied through the keyboard switches, but also converting this information into a form that the rest of the system can easily handle. To carry out this conversion the encoder has to take certain decisions.

The steps in encoding numbers

First of all we will concentrate just on finding out how the encoder generates the number signal for the display register. The number is shown as a broad arrow on *figure 2*. You can take it for granted that the signals to the controller, represented by the other broad arrow, are produced in much the same way. (These signals tell the controller when a number key is pressed, but not which one, and when an operations key is pressed such as plus, equals, multiply, etc.)

So what are the decisions taken when a keystroke is encoded into binary code? The encoder takes two steps to generate a number, and each step can be used to illustrate a different type of basic decision-making circuit. *Figure 3* shows these two decision blocks as parts of the keyboard encoder.

In the first section circuits of one type decide which number key has been pressed, according to which of the keyboard-input lines and which of the scan lines are on. The result is transmitted by turning on one of ten 'number lines' that go to the second section.

In this section another type of circuit decides which of the four lines going to the display register have to be turned on in order to transmit the number in the required binary code.

AND gates

Now let's examine the switching circuit that decides when the number 1 line from section one to section two should be switched on. *Figure 4* shows what this circuit has to do and from where it gets its information.

The circuit has the job of turning on the number 1 line whenever the key 1 is pressed on the keyboard. Remember that pressing the key 1 causes keyboard input line N to be 'on' when scan line 1 is on. None of the other keys (such as the other three shown in *figure 4*) can make both of these input lines be on at the same time. So our number one switching circuit must turn on whenever both input line N and scan line 1 are on.

You could say that this circuit is a coincidence detector because it reacts only when it finds *both* signals on *at the same time*. It's rather like a gate with a lock and a handle. Turning the key allows the handle to open the gate, or turning the handle and then the key will have the same effect. But try one without the other and the gate will not open. In fact this type of circuit is called a **gate circuit.**

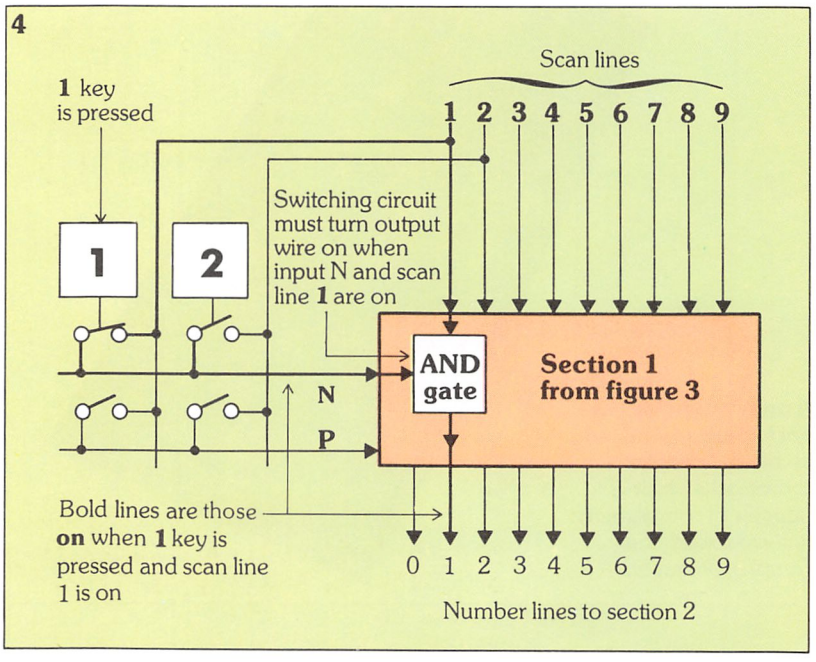
There are a number of different types of circuit which are also known as gates; this one is called an AND gate. The AND is always written with capital letters and signifies that 'this' and 'that' must happen together to make it work (turn the key AND turn the handle to open the gate).

How an AND gate works

Regardless of the actual circuit design (we will look at this later) all AND gates can be thought of as being like the circuit shown in *figure 5*. They all act like electrically controlled switches connected in series, with each switch turned on or off by a signal in a particular control line. (In series means the same current passes through both switches.)

In our example, when scan line 1 is on together with input line N, then both switches are closed and an electrical current flows from the power supply to the output line. This then allows us to know what the output will be for all the possible combinations of input signals, as summarized in the function table alongside the

4



circuit. That's all there is to it. In effect this circuit has made a decision to switch on or off the power in its output line, and has based this decision on the electrical condition of two input lines.

You have just learned a very important fact. Digging down to the very foundations of our calculator we have found one of the building-blocks that help make up all digital systems. Connected together in the right way, large numbers of these AND gates, along with a few other types of very similar circuits, are what make every digital

4. The function of the switching circuit found in section one of the keyboard encoder shows what an AND gate does.

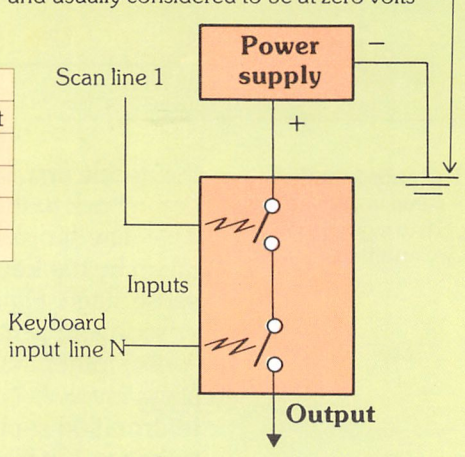
5. An imaginary switching circuit which indicates how an AND gate works, and the function table for the circuit.

5

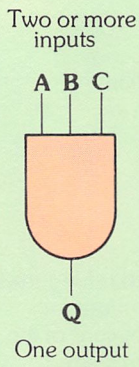
Output is on when scan line 1 and keyboard input line N are both **On**

Function table*		
Scan 1	N	Output
Off	Off	Off
Off	On	Off
On	Off	Off
On	On	On

*To be precise in defining positive and negative logic, assume **On** means higher, more positive voltage, and **Off** means lower, more negative voltage



6



Truth table for 3-input AND gate

A	B	C	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Output is **yes** (**1**) only when all inputs are 1. Output is **no** (**0**) when any one or more inputs are 0.

Truth table for circuit in fig 4. using positive logic

Scan 1	N	Output
0	0	0
0	1	0
1	0	0
1	1	1

6. The usual symbol for an AND gate and a truth table showing its precise operation for all different combinations of inputs.

system work.

Here, once more, we can recognize the pattern that will be repeated again and again in digital devices: the decision that a system is required to make is broken up and subdivided into very simple decisions, which can then be handled by very basic electronic circuits.

The standard symbol for an AND gate

It is often necessary to show several AND gates in a small space. To save having to label a number of boxes as AND gates a standard symbol is used, shown in *figure 6*. To keep the drawing simple, just the input and output lines are shown. There are power lines of course, but these are left out of the diagram.

7. A diagram of a digital circuit with three inputs and four outputs. This means that there are 128 possible combinations of input and output states.

7

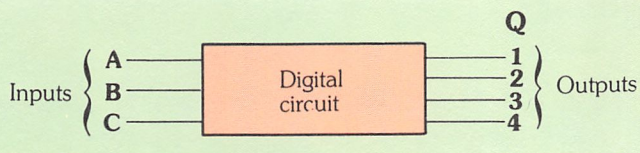


Figure 6 introduces another new point. You'll notice that the symbol has *three,* rather than two, inputs (labelled A, B and C). This is to show that an AND gate can have more than two inputs.

Truth tables

Figure 6 also shows the definition of an AND gate more precisely than before. An AND gate is actually defined in terms of the *logical meaning* of the inputs and outputs. That is in terms of the two basic bits of information a line can carry, rather than in terms of the electricity itself. As we saw earlier, we call these bits 1 ('yes' or 'true') and 0 ('no' or 'false').

So, to be precise, an AND gate is any circuit with two or more inputs and one output, whose output will be 1 only when all the inputs are 1. When one or more of the inputs are 0, the output too will be 0. The larger table in *figure 6* shows how this relates to a 3-input AND gate. It's called a **truth table**. A truth table can be described as a shorthand representation of the possible output conditions of a given circuit defined according to the possible input conditions. By circuit conditions we mean 1 or 0, on or off, yes or no, true or false, open or closed. Whatever terms are used, in the end it means current flowing or current not flowing.

Looking at the circuit shown in *figure 7* you will see that it has three inputs and four outputs. This means that there are eight possible input combinations (there are three lines each with two possible states – on and off – to give $2^3 = 8$ possibilities). On the output there are $2^4 = 16$ possible combinations. 2^3 input combinations and 2^4 output combinations give 2^7 possible input/output combinations. This means that for a three input, four output digital circuit, there are 128 possible combinations.

Obviously, drawing up a truth table which includes all possible combinations is not going to be a very concise way of defining a digital circuit. But, in effect, not every possible combination of input or output will be used by the system, and the circuitry inside the gate will be restricted to match these requirements. So the truth table that is drawn up is restricted to the combinations allowed by the circuit configuration.

Going back to *figure 7,* we could say that this circuit is controlling four separate units which have to work in the following way:
When input A is on output 1 has to be on.
When input B is on 2 has to be on.
When input C is on 3 has to be on.
When inputs ABC are on all outputs have to be on.
And when inputs A B and C are all off then output 4 has to be on.

This is a lengthy description of what the circuit does, but look in *figure 8* at how

it can be described in shorthand using a truth table. Remember that in this case 1 = ON, 0 = OFF.

Positive logic

To define this AND gate more precisely, we have to specify exactly what is meant by on and off in this particular application. In the scan-line input 'on' means 'yes, electricity is being supplied to all the switches connected to scan line 1 and to no others'. In the same way, 'on' in keyboard input N means 'yes, electricity is coming from one of the switches connected to this line'.

This method of supplying information is known as **positive logic.** The presence of voltage of a predetermined value is regarded as having a value of 1; the absence of it, or the presence of voltage of a lower value, is regarded as having a value of 0. So in *figure 4* let's assume that the voltage representing the on condition is larger than the voltage representing the off condition.

Using positive logic we can write the truth table for the AND gate in *figure 4*, simply by copying the function table from *figure 5*, writing 1 for ON and 0 for OFF. This leaves us with the truth table as shown in *figure 6*.

Negative logic

Suppose now that the circuit in *figure 5* were to be used in another application where on (high voltage) meant no or 0, and where off (low voltage) meant yes or 1. This is normally called **negative logic.** In this case the truth table would be different. The output would be 0 only when both inputs are 0. So the circuit used as an AND gate for positive logic can't work as an AND gate for negative logic.

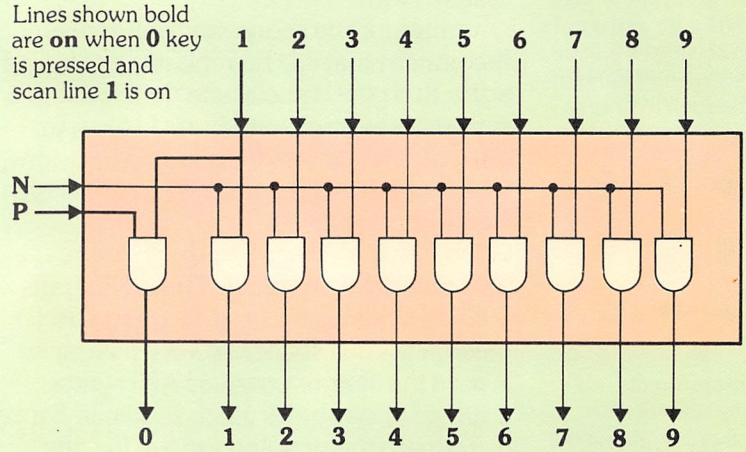
This is why, to be precise in naming the circuit shown in *figure 5*, it should be called a positive AND gate. A negative AND gate is a completely different circuit. Whenever the standard symbol for the AND gate is used to refer to a real electrical circuit, the type of logic, either positive or negative, must be specified. That way the symbol is automatically defined as a positive or a negative AND gate. (Unless specified all the circuit symbols used in this series will be using *positive* logic).

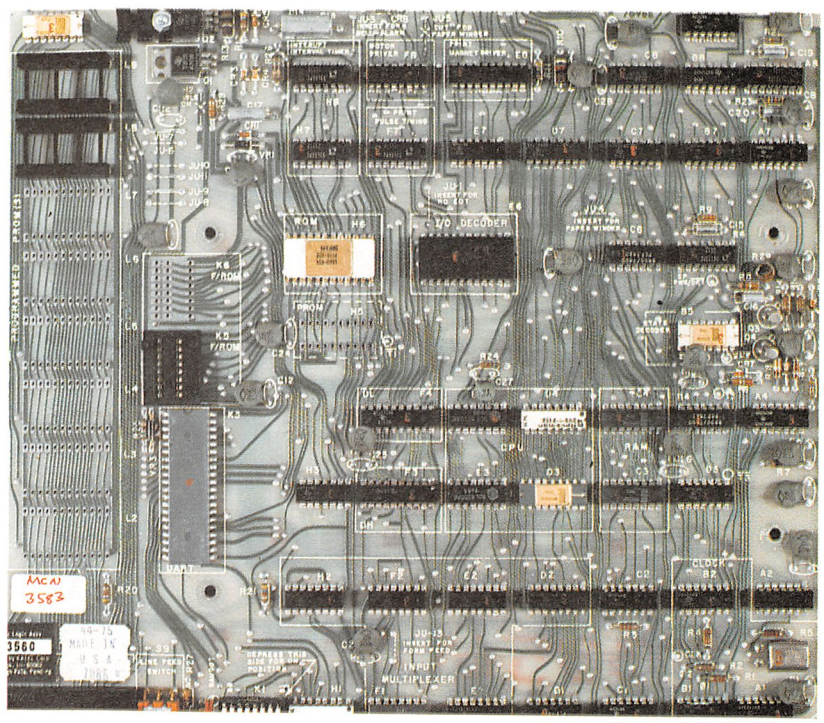
8

Inputs			Outputs			
A	B	C	Q_1	Q_2	Q_3	Q_4
1	1	1	1	1	1	1
1	1	0	1	1	0	0
1	0	1	1	0	1	0
1	0	0	1	0	0	0
0	1	1	0	1	1	0
0	1	0	0	1	0	0
0	0	1	0	0	1	0
0	0	0	0	0	0	1

8. Truth table showing the input/output combinations of the circuit in figure 7.

9





ZEFA

OR gates

Now that we have explained an AND gate and the use of the standard symbols, let's look at a diagram of the entire first section of the encoder. Each scan line has its own 2-input AND gate, as shown in *figure 9*. Each AND gate has one of its inputs connected to either key input line N or P. The outputs of the AND gates go to another type of gate, known as an OR gate.

An **OR gate** forms the second section of the encoder. Go back to *figure 3*; you will recall that the second section decides which of the four lines to turn on, to make a number for the display register. This decision is made by several OR gates. The entire circuit for encoding numbers, with both sections 1 and 2 included, is shown in *figure 10*.

The ten AND gates at the top are just the same as in *figure 9*. The OR gates to which they are linked are shown on the right of the diagram. The spear-head shape used is the standard symbol for an OR gate. (As we are using positive logic, to be precise the symbol represents a positive OR gate.)

You can get some idea of what the OR gates do by following the bold lines in *figure 10*. These are the lines which are on (1) when key 5 is pressed, generating the binary code 0101, which stands for 5.

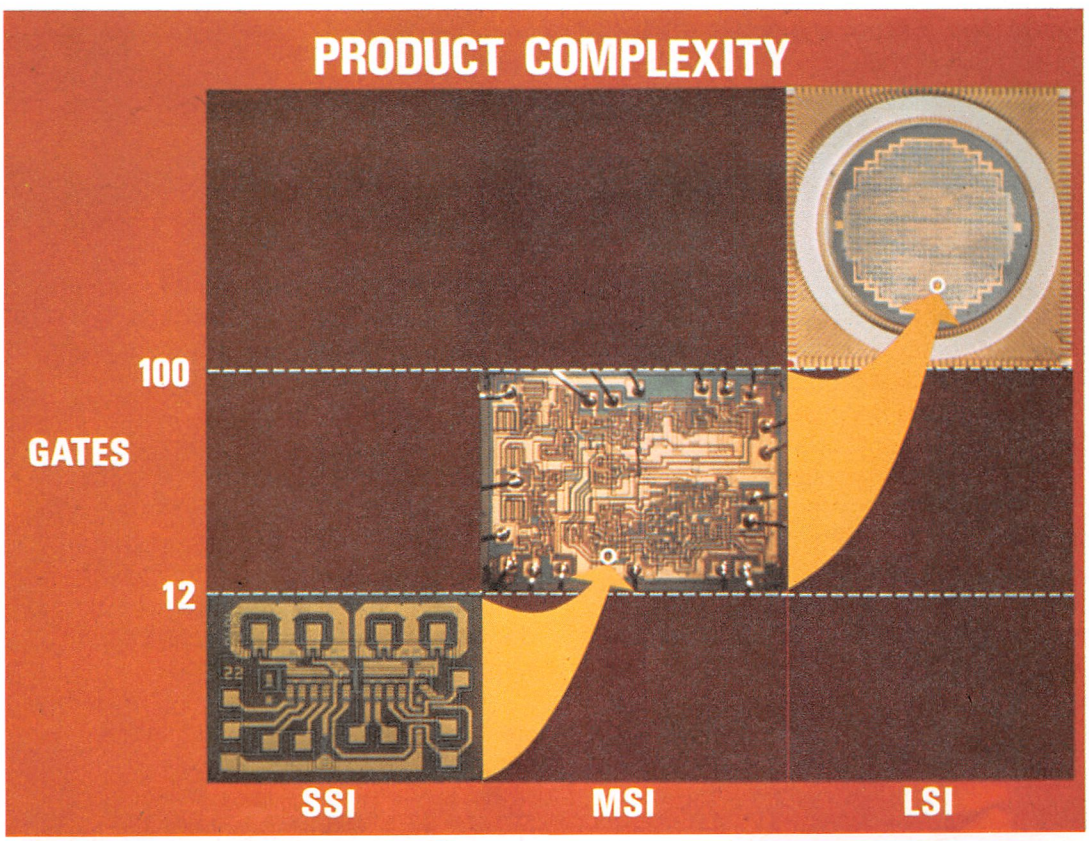
You have already seen how one of the AND gates turns on line number 5 inside the encoder. Now you can see that this line is connected to the two OR gates numbered 1 and 4. When only these two gates are active the number 5 (0101) is sent to the display register.

The function of an OR gate

Figure 11 shows that an OR gate's job is to transmit a 1 when *any* one or more of its inputs are 1. The output is 0 only when *all* the inputs are 0. This action is summarized in the truth table shown for a 3-input OR gate.

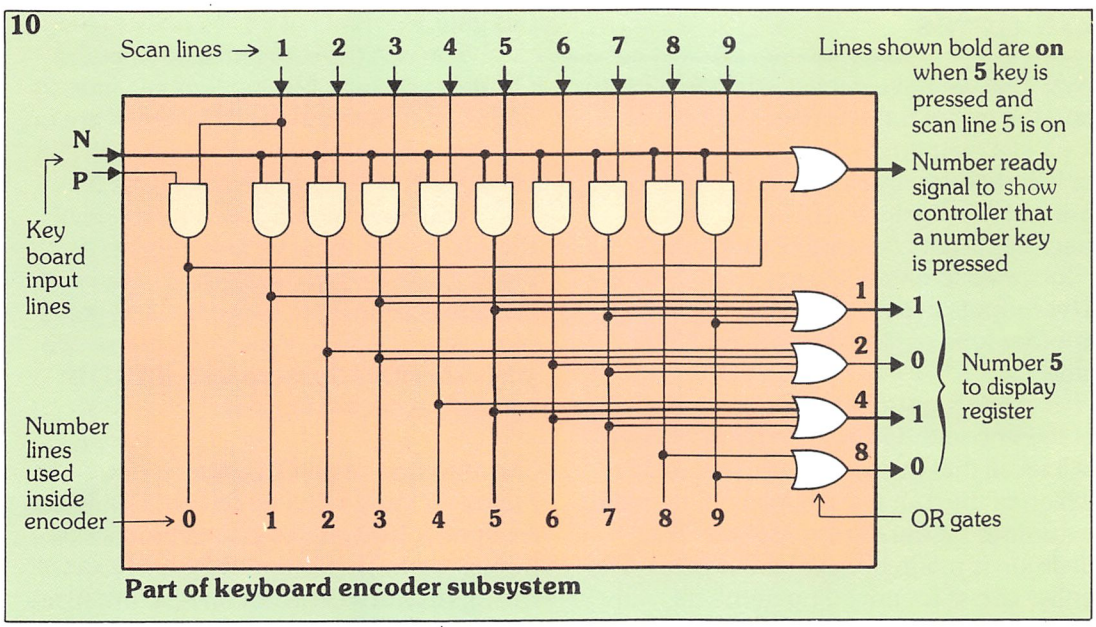
This function is given the name OR because the output is true (1) when *one or other* of the inputs is 1. To that is added the name gate. Remember the AND gate analogy of a gate with a key and a handle, and both had to be turned to open it? This time the gate has two handles and either

9. How AND gates are arranged to form section one of the keyboard encoder.



Right: the progression from Small to Large Scale Integration means that an increasing number of gates can be fitted in the same area of silicon chip.

Left: microcomputer 'logic board'; all the main components are contained on a single circuit board.



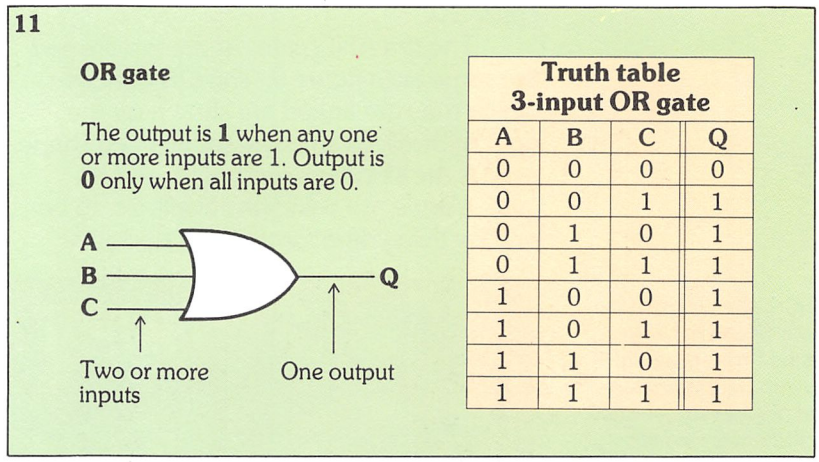
10. Complete design for section two of the keyboard encoder showing the use of AND and OR gates.

11. The standard symbol and truth table for an OR gate.

12. The operation of an OR gate represented by a switching circuit.

one OR the other can be used to open it. Thus you can open an OR gate by having just one input at 1, whilst an AND gate can only be opened when all inputs are 1.

To understand better how an OR gate works, go back to *figure 10* and look at the OR gate which turns on the binary 8 position output. This OR gate's output would be 1 whenever the input it receives from either the number 8 or 9 line is 1. These two lines have to connect with the bottom OR gate which controls the binary 8 position, because numbers 8 and 9 both have a 1 in this position when expressed in the binary code used here (ie. 1000 and 1001).



Truth table 3-input OR gate

A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The zero key

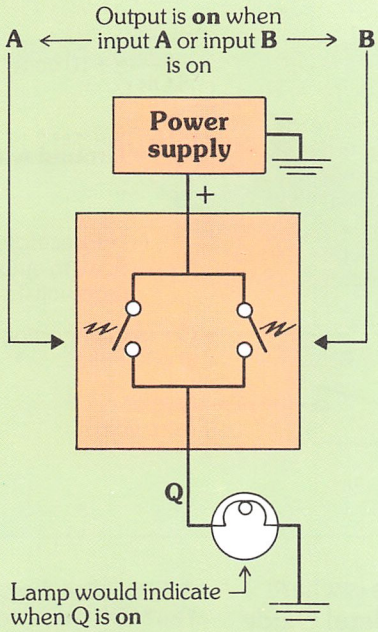
If you look closely at *figure 10* you'll see that the zero key does not go to any of the four OR gates that produce the binary numbers. This is because we don't want the zero key to activate a 1 in any of the binary number output lines. Zero in the binary code we are using is 0000. However, zero does connect to a separate OR gate at the top right hand of the diagram. This gate controls the 'number ready' signal which is passed to the controller, a signal which applies to a zero just as much as numbers 1 to 9. When it receives this signal, the controller can then go through the appropriate routines (discussed in chapter 1) and tell the display register to store the number. Remember the controller doesn't need to know exactly which number key is pressed. The output from this OR gate is 1 whenever either the zero number line is l (zero number key pressed) OR input line N is 1 (any other number key pressed).

So that is the complete design of the number-encoder network, the most important part of the keyboard encoder. The only other function of this subsystem is to detect when one of the eight operation keys (plus, minus, equals, etc.) is pressed, and to tell the controller which one. This is done in much the same way as for the numbers.

The network shown in *figure 10* may seem simple when compared with the complexity of the entire calculator, or other digital systems. But it illustrates not only the two most important gate functions, but

The corner of a single microchip showing the complex integrated circuitry contained in a tiny area.

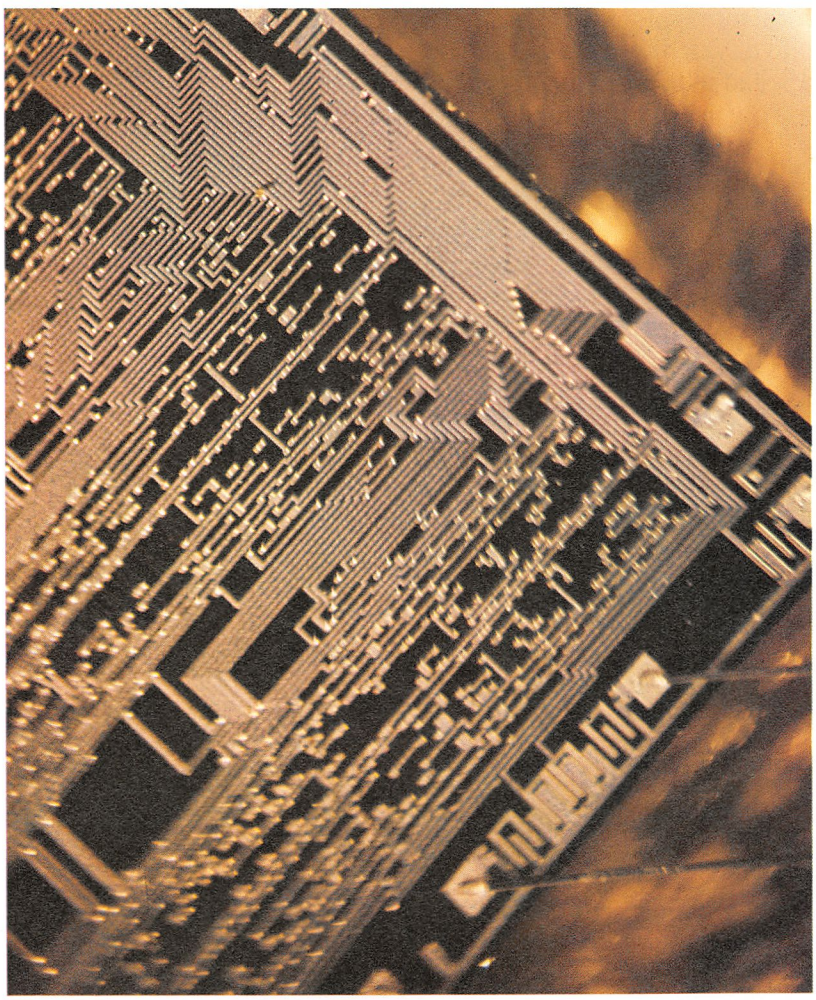
12



Function table*		
A	B	Q
Off	Off	Off
Off	On	On
On	Off	On
On	On	On

***As** in figure 5, assume
On = higher, more positive voltage
Off = lower, more negative voltage

Truth table using positive logic		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



Paul Brierley

also the operating principle of one of the most basic types of subsystem – one that handles *codes*.

This will be explored in more depth as the course progresses. You will see that this general pattern of gates – a row of AND gates followed by a row of OR gates – is repeated time and time again in many different kinds of subsystem.

How an OR gate works

As far as the gates used in the encoder are concerned, you have already seen how an AND gate works in terms of positive logic (high voltage = 1, low voltage = 0). *Figure 5* showed that this function could be thought of as electrically controlled switches, one for each input, connected in series.

Figure 12 gives a similar picture of an OR gate when it is used with positive logic. Once again there is an electrically controlled switch for each input (there can be as many inputs as you want although only two, marked A and B, are shown here). However, in this case the switches are connected in *parallel* rather than in series. So now the output would be on when *either A or B or both* inputs are on. A lamp in the output would light up when either input was on.

The function table of this circuit is shown at the top right of *figure 12*. Remember that the function table for any digital circuit shows all possible combinations of the input state.

Once again to change the function table into a truth table let 0 = Off and 1 = On. The truth table in *figure 12* is for a 2-input OR gate using positive logic.

How real gates operate

Now that we have explained the principles of how AND and OR gates can be combined to handle complicated decisions, let's carry on to see how the gates actually work in an electronic device, and how transistors are connected together to act as gates. In effect transistors take the place of the switches used in *figures 5 and 12*. Many different types of transistors can be used in gate circuits. But for now let's concentrate on the one that acts in the most similar way to the switches (and the one that is most used in calculators). This is the MOS transistor.

MOS transistors

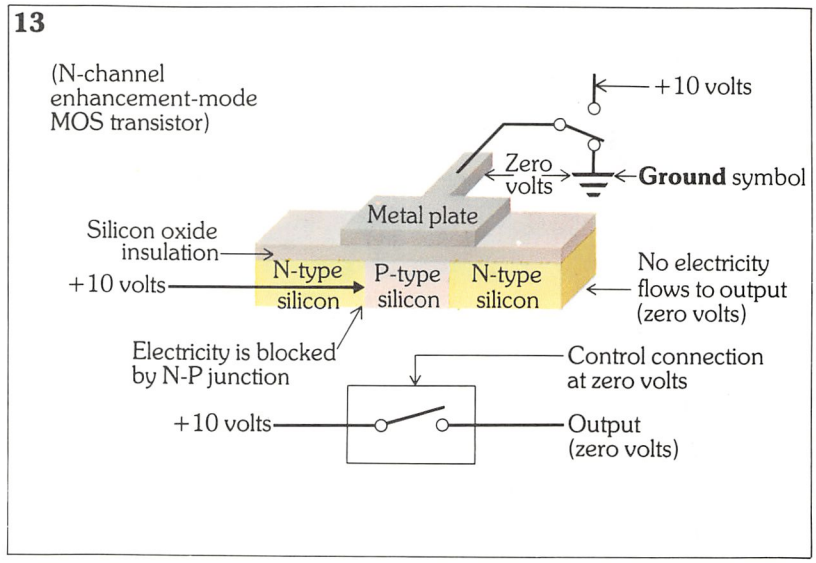
Figure 13 shows a very simplified version of an MOS transistor's internal structure. This particular type is called an n-channel enhancement mode MOS transistor. MOS transistors are very small and *figure 13* represents an area in an integrated circuit that is no bigger than a tiny speck. The diagram at the bottom of *figure 13* is a reminder that this transistor can be roughly equated to the imaginary switch we have used to help describe how AND and OR gates work.

At this point, it's not important to know about semiconductor theory in any detail. Let's just say that the main part of the transistor is a bar of silicon made of two different types: n-type and p-type. The p- and n- types were obtained by adding certain other materials to pure silicon, and each type reacts differently to electricity. Each end of the bar is n-type silicon, while the middle is p-type.

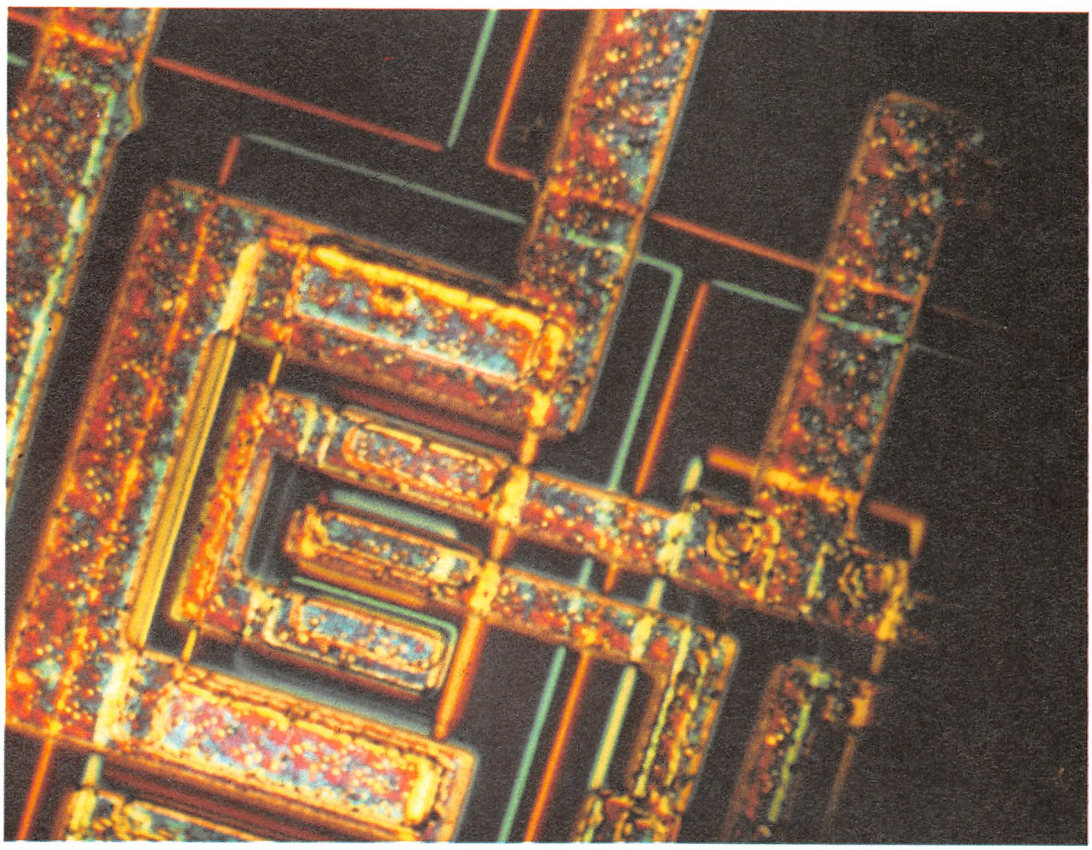
On top of the silicon bar is a layer of a glass-like substance called **silicon oxide.** This acts as an electrical insulator, so no electricity can pass this layer. Above the oxide layer is a **metal** plate. This is where the name MOS comes from – **Metal Oxide Semiconductor.** The standard symbol for an MOS transistor is shown in *figure 15.*

The two end n-types form the electrical terminals (like the connections going into a switch). If a supply of ten volts is applied to the left-hand terminal it will try to flow through the silicon to the right-hand

13



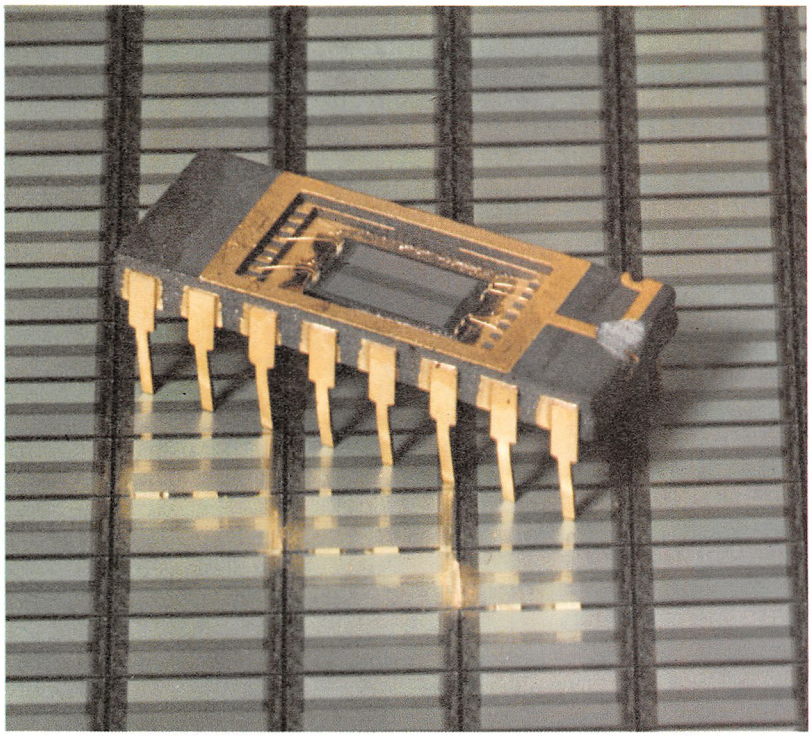
13. Simplified diagram of an MOS transistor in the 'off' state, and its representation as an electrically controlled switch.

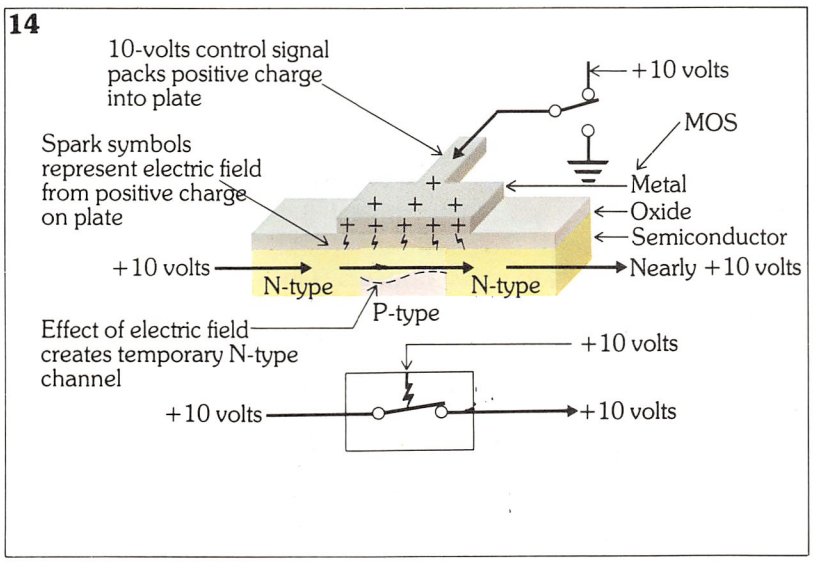


Paul Brierley

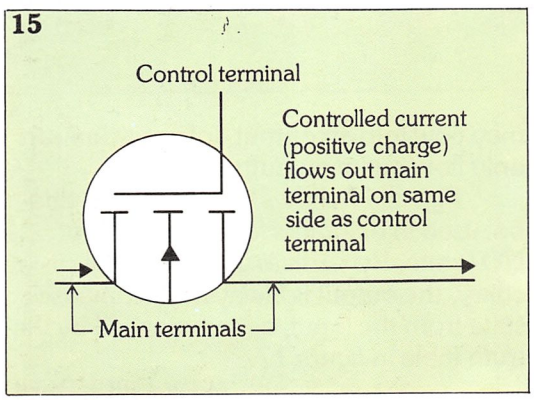
Above right: IC package with the outer casing removed to reveal the MOS semiconductor chip.

Microphotograph of a single gate on a microchip.





14. The MOS transistor now shown in the 'off' state.



15. Standard circuit symbol for an n-channel enhancement mode MOS transistor.

terminal. But where it meets the first n-p junction it is blocked. Because of the chemical structures of the two types of material, it is a basic law of semiconductor action that current (flow of positive electric charge) cannot flow across a junction from n-type material to p-type material. So the transistor is now in the off state (the switch is open).

The metal plate acts as the control terminal of the 'switch'. As the schematic diagram indicates in *figure 13* the plate is being held at zero volts by a two-way switch. Let's see how this switch applies a control signal to turn the transistor on.

How do you turn an MOS transistor on?
Figure 14 shows what happens when we flip the controlling switch to ten volts. The voltage pressure packs positive electric charge into the metal plate. Here, the charge finds itself at a 'dead end' because it cannot pass through the oxide insulation.

However, the positive charge creates an electric field, which does exert itself right through the oxide, as a magnet exerts a field through a sheet of paper to a nail on the other side. This field is indicated in the diagram by the little spark symbols. (There are no actual sparks involved – just an electric field).

This electric field has a remarkable effect on the p-type region, an effect which is one of the basic properties of semiconductors. The upper part of the p-type gap takes on the characteristics of n-type as long as the positive charge is maintained on the plate above. Thus, a temporary n-type channel is formed between the two n-type terminals, allowing current to flow, and in effect turning on the transistor. This fact is illustrated by the symbolic switch in *figure 14*.

How do you turn an MOS transistor off?
To turn the transistor off again, it's important to note that we have to flip the control switch back to zero volts again. This is because we have to provide a path for positive charge to drain out of the metal plate. If we simply turned the control switch off, cutting the plate off from any electrical contact, the positive charge would remain on the plate until it somehow leaked out, and so the transistor

would fail to turn off for a while. This fact is crucial in the design of gate circuits.

How the transistor fits into a switching circuit

There is a type of switching circuit known as an **inverter** which makes a good illustration of how the MOS transistor is used in a digital gate. The inverter (sometimes called a NOT gate) is equally as important as the other two basic building blocks we have covered, the AND and OR gates.

Figure 16 is a diagram of two MOS inverters with the output of the one on the left acting as the input to the one on the right. The switch diagrams below give a clear mental image of how the system behaves as electrically controlled switches.

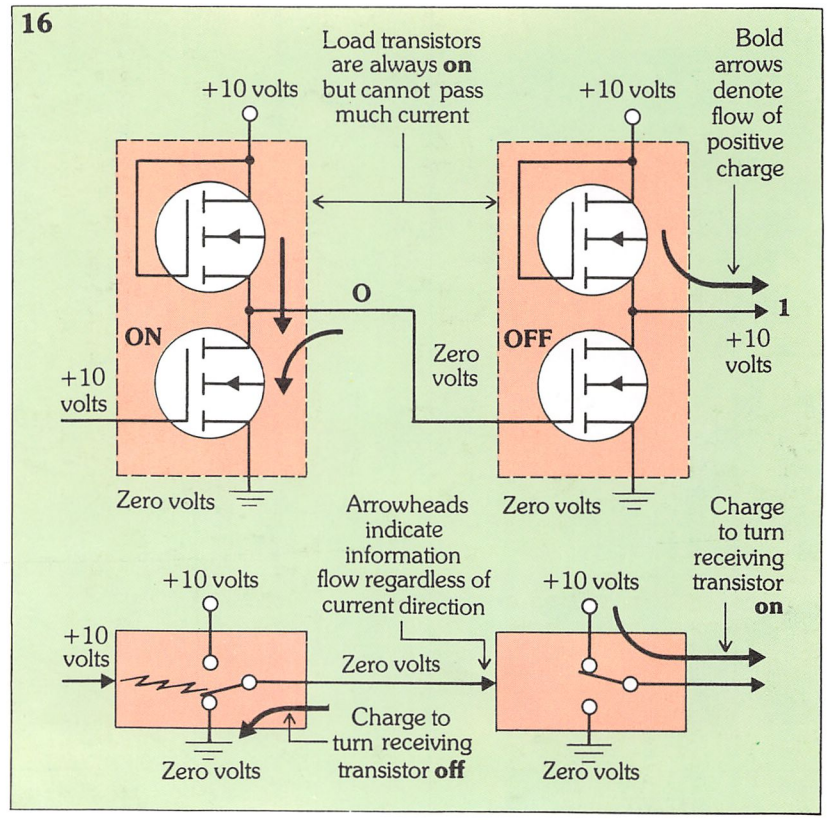
The upper transistor in each inverter (called the 'load' transistor) is specially made with a long and narrow channel between two n-type terminals. It is always kept 'on' by having its control terminal connected to the 10-volt power supply. Designed to allow only a small current to flow, it acts like a resistor by choking back most of the current.

When the lower or input transistor is off, the load transistor can supply enough current through the inverter's output to quickly charge up the control plate of the receiving transistor in the second inverter (very little current is required). But when the input transistor is on, its output is connected to ground and this drains the small amount of current from the load transistor and from the output of the inverter. In this way when the receiving transistor is on it causes the receiving transistor in the second inverter to be off.

The inverter, as its name suggests, is designed to change or invert an incoming signal to its opposite state. In our MOS circuitry on (1) is ten volts and off (0) is zero volts. The inverter's output is always the inverse of the input. In simpler terms we might say that the output is flipped over from the input.

Two-way output current

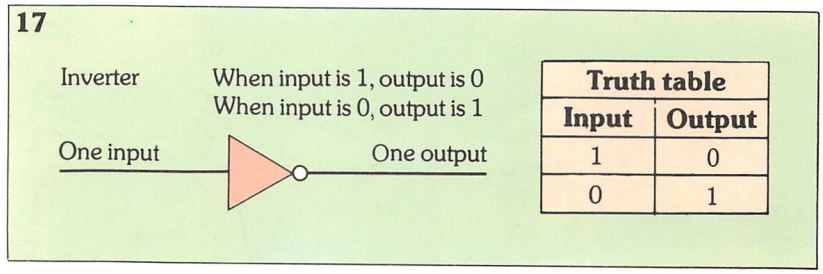
Figure 16 also shows that an MOS switching circuit must not only be able to supply electricity through its output but must also be able to drain electricity back into the output. This is so that the output can turn



16. Circuit and 'switch' diagrams for two MOS inverter circuits.

off an MOS transistor in a receiving circuit. This requirement of two-way output current applies not only to the inverter circuits but to all MOS gates, which as we will see are constructed very much alike. In fact, it applies to all electronic digital circuits. Current must be able to move both ways in all signal wires.

However, it is important to note that in digital circuits, even though electricity



17. Standard symbol and truth table for an inverter circuit.

may flow into an output, **information** can only flow out of an output.

Figure 17 shows the customary symbol used in diagrams for the inverter or NOT gate. Regardless of the internal circuitry, the output is always in the inverse state from the input, as summarized by the truth table in *figure 17*.

(continued in Part 4)